UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

Análisis y detección de bots en Twitter

Pedro Burgos Gonzalo

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Análisis y detección de bots en Twitter

Autor: Pedro Burgos Gonzalo
Tutor: Álvaro Ortigosa Juárez

junio 2021

Pedro Burgos Gonzalo
Análisis y detección de bots en Twitter

Pedro Burgos Gonzalo
C\ Francisco Tomás y Valiente nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mis padres y a mi hermano

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor Álvaro Ortigosa por las excelentes lecciones durante la carrera, por ayudarme y aconsejarme en la realización de este trabajo, y por orientarme en la elección del máster.

Además, me gustaría agradecer a toda mi familia, especialmente a mis padres y a mi hermano, por apoyarme en todo lo que hago y por la paciencia y ayuda, pues sin ellas no habría podido alcanzar esta meta.

También quiero agradecer a mis amigos de toda la vida, por todos los años que he pasado con ellos, por los buenos momentos, y también por no rendirse conmigo en los malos. Del mismo modo a mis amigos de la universidad, especialmente a Alejandro Asenjo, Iván Bartolomé y Jose Cebrián por ayudarme durante toda la carrera, y en sobre todo a Xing Xin Chen por además haber sido un excelente compañero de prácticas.

Y, por último, a mis amigos Leon y Amanda, por ser una fuente de inspiración constante, por hacerme compañía mientras escribía este trabajo, y en general, por que sé que puedo contar con ellos cuando les necesito.

Resumen

En la actualidad las redes sociales son una parte fundamental de la vida diaria de las personas. En los últimos años hemos observado como estas han tomado funciones políticas y mediáticas, y se han convertido en herramientas fundamentales para muchas empresas. No es de extrañar, por tanto, que se haya desencadenado un aumento en el pensamiento conspiranoico y la cantidad de noticias falsas distribuidas.

Numerosos estudios tratan de atajar este problema centrándose en analizar el contenido publicado, y contrastarlo con diferentes tipos de fuentes verídicas, sin embargo, esta tarea ha probado ser extremadamente compleja y acarrea un importante riesgo de resultar sesgada. Sin embargo, dado que a menudo la popularidad de este tipo de noticias es debida al uso de perfiles automatizados o bots, la tendencia en los últimos años ha sido atacar al medio de propagación de dicha información.

En este trabajo se detallarán diferentes métodos de detección de usuarios falsos, comparando sus aproximaciones y en algunos casos midiendo sus resultados. Con este fin, se empleará el dataset “cresci-2017”, desarrollado por MIB (My Information Bubble). Este dataset destaca por ser uno de los mas completos hasta la fecha en el ámbito de la detección de bots. Además, se hará uso de diferentes algoritmos y técnicas conocidas en el panorama del aprendizaje automático, tales como Random Forest (RF), Redes Neuronales (MLP), o Clustering no Supervisado (KMeans). En concreto, nos centraremos en analizar si el uso de métodos no supervisados puede ser una alternativa a los actualmente empleados.

Finalmente, se propondrán soluciones a los problemas mas relevantes de cara al desarrollo de una solución sostenible y generalizable en el tiempo, y se discutirán las posibles alternativas a explorar en el futuro.

Palabras clave

Twitter, fake news, bots, preprocesamiento, inteligencia artificial, clasificación, ingeniería de características, supervisados, no supervisados, aprendizaje automático, modelos

Abstract

Nowadays, social media is a fundamental part of daily life of people. In recent years we have observed how it has taken political and media functions, and have become tools fundamental for many companies. It is not surprising, therefore, that there has been an increase in conspiracy thinking and the amount of fake news distributed.

Numerous studies try to tackle this problem by focusing on analyzing the published content, and contrasting it with different types of truthful sources, however, this task has proven to be extremely complex and carries a significant risk of being biased. However, since the popularity of this type of news is often due to the use of automated profiles or bots, the trend in recent years has been to attack the means of spreading such information.

In this work, different methods of detecting fake users will be detailed, comparing their approaches and in some cases measuring their results. For this purpose, the “cresci-2017” dataset, developed by MIB (My Information Bubble) will be used. This dataset stands out for being one of the most complete to date in the field of bot detection. In addition, different algorithms and techniques known in the machine learning landscape will be used, such as Random Forest (RF), Neural Networks (MLP), or Clustering KMeans (KMEANS). Specifically, we will focus on analyzing whether the use of unsupervised methods can be an alternative to those currently used.

Finally, solutions to the most relevant problems will be proposed with a view to developing a sustainable and generalizable solution over time, and possible alternatives to be explored in the future will be discussed.

Keywords

twitter, fake news, bots, preprocessing, artificial intelligence, classification, feature engineering, supervised, unsupervised, machine learning, models

Índice

1 Introducción **1**

1.1 Motivación . . . 1

1.2 Objetivos . . . 2

1.3 Estructura . . . 2

2 Estado del Arte **3**

2.1 Las fake news . . . 3

2.2 La evolución de los bots . . . 4

2.3 Técnicas de clasificación conocidas . . . 4

2.4 Técnicas empleadas en el desarrollo . . . 5

2.4.1 Obtención de datos . . . 5

2.4.2 Análisis de datos . . . 5

2.4.3 Clasificación . . . 7

2.4.4 Análisis de resultados . . . 8

2.5 Herramientas empleadas . . . 10

3 Diseño y Desarrollo **11**

3.1 Metodología . . . 11

3.2 Obtención de datos . . . 11

3.2.1 Botometer . . . 12

3.2.2 MIB . . . 12

3.3 Preprocesamiento de los datos . . . 13

3.3.1 Introducción . . . 13

3.3.2 Reducción de volumen de datos . . . 13

3.3.3 Corrección de valores erróneos . . . 14

3.4 Análisis de datos . . . 15

3.4.1 Perfiles . . . 15

3.4.2 Tweets . . . 17

3.5 Extracción de características . . . 17

3.5.1 Extracción de características . . . 17

3.5.2 Dataset resultante . . . 22

3.6 Clasificación . . . 23

3.6.1 Métodos de clasificación supervisados . . . 23

3.6.2 Métodos supervisados escogidos . . . 25

3.6.3 Métodos de clasificación no supervisados 25

4 Resultados **29**

4.1 Modelos supervisados (Dataset Completo) 29

4.1.1 Precisión 29

4.1.2 Tiempo de entrenamiento 30

4.1.3 Tiempo de clasificación 30

4.2 Modelos supervisados (Mejores 10 atributos) 31

4.2.1 Precisión 31

4.2.2 Tiempos de entrenamiento 31

4.2.3 Tiempos de clasificación 31

4.3 Modelos no supervisados 32

4.3.1 Precisión 32

4.3.2 Tiempo de ejecución 33

4.4 Resumen y modelos propuestos 33

4.4.1 Modelo General 33

4.4.2 Modelos especializados 34

4.4.3 Aprendizaje no supervisado 35

5 Conclusiones y trabajo futuro **39**

5.1 Conclusiones 39

5.2 Trabajo Futuro 40

Bibliografía **44**

Apéndices **45**

A Análisis de datos **47**

A.0.1 Examen de los atributos del dataset 47

A.0.2 Correlación entre atributos del dataset 48

A.0.3 Verificación de características 49

B Resultados de modelos supervisados **51**

B.0.1 Resultados: todos los atributos 51

B.0.2 Resultados: 10 mejores atributos 52

C Resultados de modelos no supervisados **53**

C.0.1 Resultados: tras aplicar PCA 53

C.0.2 Visualización: usando el featureset “all” 54

C.0.3 Visualización: usando el featureset “users” 55

C.0.4 Visualización: usando el featureset “tweets” 56

Listas

Lista de figuras

3.1 Distribución de atributos de usuario . . . 15
3.2 Número de *followers* y de *friends* . . . 16
3.3 Función de Densidad Acumulada de *reputation* . . . 16
3.4 Número de tweets divididos por plataforma . . . 17
3.5 Visualización de los usuario del dataset mediante PCA . . . 22

4.1 Resultados con modelos supervisados y featureset “all” . . . 30
4.2 Resultados con modelos supervisados y featureset “all” (10 atributos más relevantes) . . . 32
4.3 Resultados con modelos no supervisados y featureset “all” . . . 33
4.4 Explicación y resultados obtenidos con el mejor modelo general . . . 34
4.5 Visualización predicciones KMeans (Fake Followers) . . . 36
4.6 Visualización predicciones KMeans (Traditional Spambots) . . . 36
4.7 Visualización predicciones KMeans (Social Spambots) . . . 37

A.1 Examen de atributos del dataset . . . 47
A.2 Correlación entre atributos del dataset . . . 48
A.3 Verificación de características extraídas . . . 49

B.1 Resultados empleando métodos supervisados y todos los atributos . . . 51
B.2 Resultados empleando métodos supervisados y los 10 atributos mas relevantes . . . 52

C.1 Resultados empleando métodos no supervisados y pca . . . 53
C.2 Visualización de clustering para el featureset “all” . . . 54
C.3 Visualización de clustering para el featureset “users” . . . 55
C.4 Visualización de clustering para el featureset “tweets” . . . 56

Lista de tablas

3.1 Dataset cresci-2017 . . . 13
3.2 Niveles de clasificación . . . 13
3.3 Características “Account Metadata” . . . 19
3.4 Características “Content Based” . . . 20
3.5 Características “Behavioural” . . . 20

3.6 Hiperparámetros para algoritmos supervisados 26
3.7 Hiperparámetros para algoritmos no supervisados 27

1

Introducción

1.1. Motivación

En la última década las redes sociales han evolucionado hasta tomar un papel altamente importante en nuestra sociedad, tanto es así que actualmente se han convertido en una parte fundamental de la vida de muchas personas. Si bien es cierto que las redes sociales amplifican nuestra capacidad de comunicación, no lo hacen libres de inconvenientes. El más importante de estos probablemente sea la facilidad con la que algunos usuarios confían en información de procedencia desconocida, sin contrastar su veracidad. Además, algunos estudios recientes han mostrado como exponer a usuarios a rumores no sustanciados incrementa su credulidad [3] [18]. Tanto es así que en 2013 el World Economic Forum lo mencionó como uno de los mayores peligros para la sociedad moderna [27]. En este aspecto, la red social a priori más problemática es *Twitter*, pues su modelo de comunicación se basa en fragmentos cortos de texto, por lo que es muy difícil proveer un contexto para la información comunicada.

Con el objetivo de lidiar con este problema, la comunidad científica ha realizado numerosos estudios centrados en la detección de información falsa y las denominadas *fake news* [5, 12], sin embargo, contrastar este tipo de información es una tarea problemática, que puede verse gravemente influenciada por fallos de imparcialidad. Es por eso, que, en los últimos años, los estudios se han centrado en otras alternativas. Una de las soluciones más aceptadas a este problema consiste en atacar al medio transmisor en lugar de a la información en sí.

Normalmente, los agentes interesados en difundir bulos utilizan cuentas automatizadas para aumentar la difusión. A veces incluso, se trata de redes de perfiles que interactúan entre sí, dificultando así que sean diferenciados de los usuarios legítimos [48]. Se han realizado numerosos estudios sobre el tema, caracterizando diferentes clases de bots [19] [15, 20], sin embargo, a medida que los bots aumentan en complejidad, los métodos de detección tradicionales pierden relevancia, y se vuelve imperativo desarrollar nuevas técnicas más sofisticadas. Además, actualmente no se dispone de una fuente de datos consistente y sostenida en el tiempo, en su lugar, se trata de pequeñas contribuciones de algunos equipos de investigadores, lo que dificulta gravemente el desarrollo de técnicas de

detección apropiadas.

Pese a que existen varias técnicas con diferentes grados de efectividad, la mayoría de los estudios hasta la fecha se centran en distinguir si la cuenta es controlada por un humano o no, pero no en el tipo de bot del que se trata [33, 42]. Esto supone que, en caso de utilizar los modelos propuestos para eliminar cuentas de la plataforma, se afectaría también a cualquier tipo de cuenta automatizada, independientemente de sus intenciones. Por estas razones, es imperativo desarrollar métodos consistentes de detección de perfiles falsos, que no solo sean escalables fácilmente con el tiempo, sino que sean consistentes y se centren específicamente en la detección de perfiles cuyo comportamiento suponga un efecto negativo.

1.2. Objetivos

Este trabajo se centra en resumir y explorar diferentes técnicas de análisis y aprendizaje automático que permitan diferenciar perfiles legítimos de perfiles falsos, con el fin de fin de poder atajar el problema de la difusión de *fake news*. En concreto, en este trabajo se centra en analizar cuentas de la red social Twitter, debido a que es una de las más utilizadas actualmente, especialmente en el ámbito político, por lo que la transmisión de noticias falsas es especialmente preocupante.

Los objetivos generales del trabajo son los siguientes:

O-1.– Considerar diferentes técnicas de obtención datos.

O-2.– Explorar diferentes técnicas de aprendizaje automático para la clasificación de usuarios.

O-3.– Comparar el rendimiento de diferentes métodos de clasificación de usuarios.

O-4.– Encontrar y proponer formas de lidiar con el problema de la detección de bots de ahora en adelante.

1.3. Estructura

A continuación, se detalla la estructura que va a seguir el presente trabajo:

1.– **Estado del Arte:** En este capítulo se introducirán los métodos existentes para clasificar perfiles de twitter, las diferentes formas de obtener datos de usuarios en redes sociales y cualquier otra información relevante para la comprensión del presente trabajo.

2.– **Diseño y Desarrollo:** En este capítulo se explicarán el diseño y metodologías empleados, tanto para la la captura de datos, como para la subsecuente clasificación.

3.– **Resultados:** En este capítulo se evaluarán los resultados obtenidos y se compararán con los publicados en diferentes estudios de índole similar.

4.– **Conclusiones y trabajo futuro:** Por último se comentarán los resultados obtenidos y sus implicaciones. Además se explicarán las posibles alternativas a explorar en futuras investigaciones o trabajos.

2

Estado del Arte

2.1. Las fake news

En los últimos años, se ha podido observar como el uso de las redes sociales ha pasado de ser un mero entretenimiento, a ser capaz de influenciar corrientes políticas. Tanto es así que la mayoría de los partidos políticos han comenzado a utilizan este medio en mayor o menor medida, con el fin de aumentar su alcance mediático. No es de extrañar, por tanto, que la difusión de información falsa haya consecuentemente aumentado consecuentemente.

Debido a la facilidad de propagación de la información en este tipo de medios de comunicación, las denominadas *fake news* se han convertido en un problema muy presente, llamando la intención de usuarios, investigadores y gobiernos por igual.

Resulta especialmente interesante destacar el caso de Twitter, una de las plataformas más usadas en el ámbito político y mediático. Por esta razón, es también sistemáticamente abusada mediante perfiles automatizados o *bots* para difundir información engañosa. Como se detalla en [36], "las *fake news* se extienden más rápido y más ampliamente que la verdad en las redes sociales". Además, ha sido probado como, extender este tipo de información sirve intereses que alcanzan desde, influenciar el panorama político [43], como provocar cambios en el mercado de valores [37]. Ejemplos de esto son, el análisis de hashtags en relación con el *Brexit* [22] o las elecciones generales en 2019 en España [31]. Las noticias falsas, en definitiva, se aprovechan del denominado sesgo de confirmación [32], que se ve propiciado por la naturaleza de las redes sociales en si.

Existen diferentes métodos de detección de noticias falsas, algunos de los cuales incluyen, diferentes variantes de procesado lingüístico de textos o análisis de redes. En [12] se describe además el concepto de *trust* para identificar fuentes de información creíbles. Sin embargo, clasificar este tipo de noticias resulta extremadamente desafiante, debido a diferentes factores como el volumen y velocidad a la que se crean o las limitaciones que las diferentes plataformas imponen en el acceso a sus datos [36].

2.2. La evolución de los bots

Como ya se ha mencionado, las *fake news* a menudo son distribuidas utilizando redes de perfiles automatizados, con el objetivo de aumentar su influencia y alcance. Sin embargo, pese a que este es un fenómeno que lleva sucediendo desde hace muchos años, continúa siendo un problema difícil de resolver.

Con el tiempo, a medida que han aparecido nuevos métodos de detección de bots más eficaces y complejos, los bots también han ido evolucionando para adaptarse. Numerosos estudios ilustran el paso de los denominados bots tradicionales y seguidores falsos, a modelos capaces de mimetizarse con los usuarios genuinos [15].

Actualmente, numerosos estudios se enfocan en clasificar diferentes tipos y subtipos de bots, de acuerdo a su intención y comportamiento. Probablemente uno de los más prominentes sea el presentado por R. Gorwa y D. Guilbeault [20], que destaca la presencia de *Cyborgs*, inicialmente descritos en [11], un tipo de bots que exhiben una mezcla entre automatización y actuación humana, y que constituyen uno de los grandes retos a la hora de la detección. Clasificaciones alternativas pueden verse en como el conducido por Oentaryo [33]. Existen algunos trabajos que destacan la importancia de comprender el propósito de los bots en la tarea de la detección [19, 42].

2.3. Técnicas de clasificación conocidas

A grandes rasgos podemos diferenciar cuatro tipos de técnicas de clasificación. A continuación, se listan algunas de las técnicas más importantes en cada una de ellas:

- **Análisis de Metadatos:** Análisis de la información básica de los perfiles.
  - **Codificación de usuarios como secuencias de ADN:** [14]
- **Análisis de Contenido:** Análisis del contenido publicado por los perfiles
  - **Uso de medidas de diversidad:** [25].
  - **Análisis del lenguaje empleado por los usuarios:** El principal inconveniente de este método es una dependencia del idioma empleado. Sin embargo, hay algunos trabajos que tratan de mitigar este hecho [28, 45].
- **Análisis de Comportamiento:** Análisis de las acciones de los perfiles
  - **Análisis estructural de las interacciones entre usuarios:** A menudo utilizando grafos en *Neo4j* [4, 26]. Presenta graves inconvenientes a la hora de construir un dataset adecuado, debido a que gran parte de los bots etiquetados en otros estudios han sido inhabilitados.
  - **Análisis temporal de las interacciones entre usuarios:** [7, 8, 10, 30]

Además, cabe destacar la existencia de diversos estudios que **combinan aspectos de varias categorías** [13, 15, 47], o que hacen uso de **componentes aislados** especializados en analizar cada uno de dichos aspectos [11].

2.4. Técnicas empleadas en el desarrollo

A continuación, se explicarán brevemente una serie de alternativas y técnicas que posteriormente serán empleadas en Diseño y Desarrollo.

2.4.1. Obtención de datos

El primer paso para la implementación de métodos de clasificación consiste generalmente en la obtención de datos. Dependiendo del problema que se plantee, y el tipo de clasificador que se pretenda implementar se optará por una determinada vía de obtención de datos. En lo relativo al problema sobre el que trata este trabajo, se consideraron diferentes alternativas:

- **Uso de la API oficial:** Twitter provee una API oficial, mediante ella, tendremos acceso a perfiles públicos de la red social, así como a toda posible información relevante. Sin embargo, el principal inconveniente de esta herramienta son las limitaciones que impone la propia compañía sobre su uso, restringiendo altamente la cantidad de peticiones que permiten realizar en un corto periodo de tiempo.
- **Uso de una herramienta 3rd party:** Existen herramientas de terceros como *Twint*, que sirven para recabar datos de perfiles actuales de forma masiva. En concreto, la principal ventaja de *Twint* es que, gracias a que no utiliza la API oficial, no está sujeta a sus limitaciones. Sin embargo, se plantea un importante inconveniente a la hora de anotar cada usuario como auténtico o bot, pues la única alternativa es que se haga de forma manual.
- **Crowdsourcing:** Algunos estudios utilizan técnicas de Crowdsourcing para recabar datos de forma pasiva [15], gracias a esto, se permite que personas anónimas puedan reportar o revisar diferentes cuentas, distinguiéndolas entre auténticas y bots. El principal inconveniente de este tipo de procedimiento es que requiere mucho tiempo y alcance para ser capaz de alcanzar un volumen de datos suficientemente alto y diverso.
- **Datasets existentes:** Se pueden reutilizar datasets creados por otros investigadores para desarrollar nuestro estudio. Entre otros, como los que se pueden encontrar en *Botometer* [17], destaca, por su tamaño y variedad, el dataset generado por el MIB, como parte del estudio de Stephano Cresci y sus compañeros [15]. Cabe destacar que, a menudo, este tipo de datasets están compuestos a partir de datos tomados a través de la API de twitter, de forma que dichos datasets serán normalmente compatibles entre si.

2.4.2. Análisis de datos

Previamente a la clasificación es recomendable realizar una fase de análisis de datos. Esta etapa nos permite inspeccionar la información con la que trabajamos, descubrir posibles errores, y transformar dicha información de acuerdo con nuestras necesidades. Además, nos encargaremos de unificar el formato de los datos empleados de cara la clasificación, facilitando así futuras ampliaciones del trabajo.

Extracción de atributos

Debido a que la información generalmente se encuentra de forma plana, directamente obtenida desde la API de *Twitter*, es necesario transformarlos a una forma en la que sean válidos. Según el clasificador que se vaya a utilizar, tener en cuenta unos u otros factores tendrá un determinado impacto en el rendimiento. Mediante el proceso de *Feature Extraction*, extraeremos atributos numéricos derivados de los datos originales.

Principal Component Analysis

PCA es una técnica que permite reducir la dimensionalidad de un conjunto de datos. Para ello, se calculan los denominados "Componentes principales" a partir del resto de los datos, esto es, un conjunto de variables no correlacionadas que maximizan la varianza. Esta técnica es especialmente útil para reducir el tiempo de ejecución, pero también sirve facilita la visualización de los datos cuando se usa para reducirlos a 2 o 3 dimensiones.

Normalización / Estandarización

Se trata de dos técnicas similares de preprocesamiento de datos, consistentes en escalar los valores numéricos de los datos a un rango común. A menudo, el uso de estas técnicas mejora significativamente el rendimiento de determinados algoritmos de Clasificación. Su uso, por tanto, depende del tipo de clasificador que pretendamos emplear. La diferencia entre ambos procedimientos radica en la escala, por una parte, normalización transforma cada variable independientemente a un rango [0-1], mientras que estandarización las escala de forma que el conjunto de los datos tenga media igual a 0 y desviación típica igual a 1.

Detección de outliers

Se usa el termino *outliers* para referirse a valores suficientemente alejados del resto de valores del dataset para no ser considerados válidos. Estos valores no siguen la misma distribución que los demás, y por tanto, a menudo es recomendable eliminarlos previamente de cara a construir un clasificador robusto. Existen diferentes métodos para su detección, tales como *Eliminación mediante percentiles*, *Isolation forest* o *Covarianza robusta*.

2.4.3. Clasificación

A continuación, se explicarán las diferentes técnicas de clasificación empleadas en la realización de este estudio.

Hyperparameter tuning

Se denomina ajuste de hiperparámetros a el conjunto de técnicas que permiten encontrar hiperparámetros que optimizan el rendimiento de un determinado clasificador. A menudo dichos hiperparémetros varían ampliamente en función del problema a resolver, por lo que es recomendable realizar pruebas con diferentes valores. Existen varios métodos de ajuste de hiper parámetros tales como *Random Search* o *Grid search*

Cross Validation (K-Fold)

Es una técnica de evaluación de resultados durante el entrenamiento de un clasificador. Esta técnica nos permite asegurar que dichos resultados son independientes de la partición de datos elegida para entrenar. Mediante esta técnica, se divide el conjunto de datos de entrenamiento en varios *folds* y se realiza un entrenamiento con cada subconjunto de K-1 folds. Al final se calcula la media entre los resultados obtenidos.

Sobreajuste

El sobreajuste es el efecto de entrenar en exceso un clasificador, de forma que se vuelve demasiado específico y no tolera variaciones en el problema. Para evitar que esto suceda, pueden utilizarse diferentes técnicas entre las que cabe destacar la ya mencionada *Validación Cruzada*.

K Nearest Neighbors

Se trata de un método de clasificación no-paramétrico que determina la clase a la que pertenece un determinado elemento en función de sus *K* vecinos más cercanos.

Se debe tener en cuenta que, al ser un método basado en distancias para la clasificación, será extremadamente importante tener en cuenta la escala de los atributos, por esto, a menudo, aplicar normalización previamente mejora ampliamente los resultado.

Random Forest

Este método hace uso de conjuntos de árboles de decisión generados aleatoriamente y agregados mediante una técnica similar al *Bagging*. Cada árbol se centrará en un subconjunto aleatorio de atributos, y separará los datos recursivamente mediante las menores divisiones posibles. De esta forma,

el conjunto de árboles no será correlacionado, solventando así el problema de ajuste excesivo que a menudo se da en árboles de decisión tradicionales.

Multilayer Perceptron

Se trata de un tipo de red neuronal consistente en la combinación de varios perceptrones simples en una topología de capas. Gracias a que posee una o más capas ocultas este algoritmo es capaz de desempeñar clasificación de problemas en los que los datos no son linealmente separables. Cabe añadir que, de forma similar a otros tipos de redes neuronales, para su entrenamiento, hace uso de una técnica denominada *Backpropagation*. Este tipo de red neuronal, aunque sencillo, resulta especialmente útil en problemas de clasificación de atributos numéricos, sin embargo, puede presentar problemas si no se tiene en cuenta la escala de los diferentes atributos.

Support Vector Machines

Las *Support Vector Machines* se basan en la estimación de un hiperplano que divida los datos construidos a partir de los diferentes Vectores Soporte. A menudo esta técnica se usa en combinación con funciones kernel que permitan aumentar la dimensionalidad de los datos de entrada, de forma que estos sean linealmente separables en el nuevo espacio dimensional.

Además, cabe destacar que la implementación SVC de sklearn que se va a utilizar, emplea la técnica *one versus one* para llevar a cabo clasificación multiclase.

KMeans

A diferencia de los métodos explicados anteriormente se trata de un método no supervisado. Esto es, trabaja con datos carentes de etiquetas. Esto facilita la obtención de los mismos, dado que no se requiere una fase de etiquetado a menudo manual. Sin embargo, este algoritmo por si mismo no es capaz de clasificar datos, sino que en su lugar detecta agrupaciones (*clusters*) de los mismos. Existen diferentes técnicas para etiquetar *clusters*, que serán descritas más adelante en Diseño y Desarrollo.

2.4.4. Análisis de resultados

Existen multitud de métodos para el análisis estadístico de resultados, algunos de los más relevantes son los siguientes.

Curva ROC

Un método muy extendido para medir el rendimiento de clasificadores binarios. Representa la tasa de verdaderos positivos frente a falsos positivos. Además, midiendo el área bajo la curva podemos

fácilmente comparar varios clasificadores en términos absolutos.

Feature Selection

Se denomina así al proceso de eliminación de atributos poco representativos para un conjunto de datos determinado. Existen diferentes técnicas que, como es esperado, varían en términos de efectividad en función del problema dado.

Reducir el número de atributos permite mejorar el tiempo de ejecución de la mayoría de algoritmos, así como, en algunos casos, mejorar su precisión gracias a la eliminación de ruido.

Shapley Values y SHAP

El **Valor de Shapley** es un método de distribución de riquezas en la teoría de juegos cooperativos. En concreto, determina como distribuir un denominado "pago" entre varios "jugadores" de forma equitativa. En el ámbito del aprendizaje automático, se puede utilizar para calcular la contribución de cada atributo al resultado de una predicción, determinando así, que atributos tienen más peso en la toma de decisiones.

SHAP ("SHapley Additive exPlanations" [29]) es un método para explicar el comportamiento de un modelo, basándose en diferentes técnicas, entre las que cabe destacar el método de los valores de Shapely del que recibe su nombre. Sus creadores, Lundberg and Lee (2016) lo propusieron como una aproximación unificada a la descripción del resultado de cualquier modelo de aprendizaje automático. Se trata de una herramienta moderna que esta ganando mucha relevancia actualmente por su versatilidad y accesibilidad.

Comparación de algoritmos de clustering

Cuando trabajamos con algoritmos no supervisados, que no soportan la clasificación de los datos, debemos adoptar métricas diferentes para comparar sus resultados. Existen numerosas alternativas, aunque cabe destacar las siguientes tres, que se diferencian por no requerir etiquetas correspondientes a clusters.

- **Silhouette coefficient:** Esta métrica describe cuan definidos son los clusters generados, a mayor valor, mejor definición [40].
- **Calinski-Harabasz Index:** Esta métrica describe la dispersión de los valores en cada cluster frente a la dispersión de los clusters entre sí. Cuando más alta puntuación se alcance, mejor será el rendimiento del modelo [6].
- **Davies-Bouldin Index:** Este índice índica la similitud media entre clusters, entendiendo como similitud una comparación entre la distancia entre clusters y el tamaño de los clusters en si mismos [16].

2.5. Herramientas empleadas

Existen múltiples variantes de las técnicas detalladas anteriormente, tanto es así que existen diferentes herramientas y librerías orientadas al ámbito del aprendizaje automático y la ciencia de datos. Algunas de las más relevantes, que han sido empleadas en este proyecto se listan a continuación:

- **Scikit Learn:** También conocida como *SKLearn*, es una de las librerías de Python más utilizadas en el ámbito del aprendizaje automático. Esta librería implementa herramientas de diferentes tipos de clasificadores y métodos de tratamiento de datos [35].
- **Matplotlib:** Es una librería de Python diseñada para la visualización de datos y creación de gráficos [23].
- **Seaborn:** Se trata de otra librería similar a *Matplotlib* que permite realizar un análisis exhaustivo de datos. En concreto destaca su capacidad para visualizar fácilmente posibles correlaciones en los datos [44].
- **Shap:** Esta es otra librería que hace uso de *Matplotlib*. En concreto, permite visualizar, de forma sencilla, resúmenes o explicaciones sobre modelos de aprendizaje automático [29].
- **Pandas** / **Numpy:** Estas librerías son a menudo usadas en combinación, pues implementan tipos de datos y métodos para la manipulación de matrices y conjuntos de datos [21, 34].

3

Diseño y Desarrollo

3.1. Metodología

Para el desarrollo de este trabajo, se ha decidido optar por la metodología que se describe a continuación.

1.– **Obtención de datos** Se han considerado las posibles alternativas a la obtención de datos de la red social Twitter.

2.– **Preprocesamiento de los datos** Se ha sometido a los datos a un tratamiento previo a la clasificación, en esta fase se reducirá el volumen del dataset, se unificará el formato de cada fragmento, y se podrán solventar solventar problemas derivados de la falta o incompletitud de ciertos valores.

3.– **Análisis de datos** Se ha realizado un análisis previo de la información, con el objetivo de verificar la integridad de la misma, así como para explorar y visualizar la estructura de esta.

4.– **Extracción de características** Se ha codificado un módulo extractor de atributos, capaz de derivar características a partir de los datos planos que se encuentran en el dataset. De esta forma, se ha construido un nuevo conjunto de datos con dichas características.

5.– **Clasificación** Se han probado y comparado diferentes técnicas de clasificación sobre el nuevo dataset, para ello, se han empleado diferentes conjuntos de hiperparámetros y se han medido, entre otros factores, rendimiento en términos de precisión y tiempo.

3.2. Obtención de datos

Como ya se ha expuesto en 2, se han tenido en cuenta diferentes alternativas a la hora de obtener datos sobre los que basar nuestros clasificadores.

En primer lugar, se han realizado pruebas de concepto utilizando la API oficial de Twitter, y *Twint*. Sin embargo, ambas han sido descartadas debido a la difícil labor de etiquetado de los datos.

Algo similar sucede con la alternativa de *crowdsourcing*, pues, si bien ha demostrado ser efectiva el pasado, no se encontró viable para este trabajo.

Por tanto, finalmente se ha optado por hacer uso de datasets preexistentes, a menudo creados para su uso en otros estudios. A continuación, se detallan algunos de los datsaets tenidos en cuenta.

3.2.1. Botometer

Botometer (también conocido como BotOrNot [17]), es un servicio que permite analizar perfiles de twitter de forma interactiva para detectar actividad sospechosa. Este proyecto ha sido desarrollado por el *Observatory on Social Media (OSoMe)* en colaboración con el *Network Science Institute (IUNI)* en la Universidad de Indiana.

En su página web, además, han publicado una lista de diferentes tipos de datasets que contienen información sobre perfiles de twitter. Sin embargo, la mayoría de estos datasets solamente contienen perfiles correspondientes a bots, con lo que sería necesario coleccionar y anotar manualmente perfiles de usuarios legítimos.

3.2.2. MIB

My Information Bubble es un proyecto desarrollado por el *Institute of Informatics and Telematics (IIT)* en colaboración con el *Institute for Advanced Studies Lucca (IMT)*. Además de realizar diversas publicaciones en la materia de la detección de bots en redes sociales, han creado el que posiblemente es el dataset más completo hasta la fecha [15]. Este dataset fue conformado en 2016, parcialmente mediante contribuciones de usuarios a través de la plataforma *CloudFlower*, y ha sido empleado en varios estudios hasta la fecha. Este dataset fue publicado en una investigación que establecía la aparición de nuevos tipos de bots más complejos que denominaba *Social Spambots*. Estos se diferenciaban por ser más activos y replicar el comportamiento de usuarios reales en contraposición a los clásicos *Fake Followers* y *Traditional Spambots*.

Se ha decidido utilizar esta base de datos para el presente trabajo, debido a que es relativamente reciente en comparación con otros datasets de índole similar, y contiene una combinación bastante variada de perfiles.

Cabe destacar que la versión de este dataset que podemos encontrar en la página de *Botometer* esta incompleta, específicamente el fragmento *genuine accounts*, por lo que ha sido necesario solicitar la variante en formato SQL al *IIT* para posteriormente transformarlo a CSV mediante la misma herramienta usada por sus creadores (*phpMyAdmin*).

Los contenidos de este dataset se pueden observar en la tabla 3.1:

Por otra parte, la estructura física del dataset consta de un directorio por fragmento, en el cual se pueden encontrar los ficheros *users.csv* y *tweets.csv*. Estos ficheros contienen la información obtenida de la API oficial de twitter, relativa al *user object* y a los diferentes *tweet objects*. Ambos ficheros pueden ser relacionados entre sí mediante los campos *user_id* y id respectivamente. Sin embargo, se debe puntualizar que los fragmentos: *traditional spambots #2*, *traditional spambots #3* y *traditional spambots #4* carecen de datos sobre tweets, conteniendo solamente datos de perfiles. Esto se debe a

Nombre	Descripción	Perfiles	Tweets	Año
genuine accounts	cuentas verificadas que son operadas por humanos	3,474	8,377,522	2011
social spambots #1	*retweeters* de un candidato político	991	1,610,176	2012
social spambots #2	*spammers* de aplicaciones móviles de pago	3,457	428,542	2014
social spambots #3	*spammers* de productos en oferta en Amazon.com	464	1,418,626	2011
traditional spambots #1	set de *spammers* usado para entrenamiento por C. Yang, R. Harkreader, and G. Gu.	1,000	145,094	2009
traditional spambots #2	*spammers* de URLs fraudulentas	100	74,957	2014
traditional spambots #3	perfiles automatizados que hacen *spam* de ofertas de trabajo	433	5,794,931	2013
traditional spambots #4	perfiles automatizados que hacen *spam* de ofertas de trabajo	1,128	133,311	2009
fake followers	perfiles que inflan el número de followers de otras cuentas	3,351	196,027	2012

Tabla 3.1: Contenidos del dataset cresci-2017 [15]

que los autores del estudio no los consideraron relevantes en su trabajo.

3.3. Preprocesamiento de los datos

3.3.1. Introducción

A la hora de realizar cualquier estudio en el ámbito del aprendizaje automático o la ciencia de datos, es recomendable llevar a cabo una fase de preprocesamiento previa a la clasificación. Esta etapa permite ajustar los datos a las necesidades del problema en cuestión, así como corregir posibles errores derivados de la ausencia de valores o la presencia de valores incorrectos.

3.3.2. Reducción de volumen de datos

Como se detalla en [2], llevar a cabo una clasificación con toda la información del usuario y su actividad es un proceso excesivamente lento, y en caso de usar la API oficial para recabar datos, requiere la realización de demasiadas peticiones. En dicho estudio se proponen los siguientes niveles de clasificación:

Tier	Description	Focus	Collect / process Time per 250 Accounts	# of Data Entities (i.e. Tweets)
Tier 0	Tweet text only	Semantics	N/A	1
Tier 1	Account + 1 Tweet	Account Meta-data	1.9 seconds	2
Tier 2	Account + Timeline	Temporal Patterns	3.7 minutes	200+
Tier 3	Account + Timeline + Friends Timeline	Network Patterns	20 hours	50.000+

Tabla 3.2: Niveles de clasificación de datos propuestos en [2]

En este trabajo, se ha considerado apropiado, sin embargo, explorar la posibilidad de un nivel intermedio entre Tier 1 y Tier 2. De esta forma, se han clasificado los diferentes tweets en base a su autor, y posteriormente se han ordenado temporalmente. Finalmente, se han limitado las dimensiones de cada grupo de tweets a un máximo de 50. Este número ha sido elegido en base al tiempo requerido para realizar la posterior clasificación.

Gracias a esta aproximación, se podrá realizar un análisis de patrones temporales y comportamiento sin necesidad de disponer de la *timeline* completa del usuario.

3.3.3. Corrección de valores erróneos

El primer problema que se ha encontrado a la hora de trabajar con este dataset ha sido la inconsistencia de sus diferentes partes. Se han aplicado las siguientes modificaciones al dataset original:

1.– **Homogeneizar fechas:** Debido a que algunos fragmentos del dataset difieren en el formato empleado para las fechas y timestamps, ha sido necesario transformar dichos campos a un formato común.

2.– **Eliminar valores erróneos:** Se ha detectado que el dataset contiene algunas entradas de tweets erróneas, en las que, debido a un error de escapado de caracteres especiales, las columnas se encuentran desplazadas e incompletas. Así, se ha procedido a realizar un filtrado de las entradas del dataset, eliminando dichos registros erróneos. Estas entradas pueden reconocerse por que el cambo *user_id* tiene valor *NaN*.

3.– **Añadir atributo "platform":** Como se detalla en el trabajo [11], la automatización de perfiles de twitter se lleva a cabo a través de herramientas 3rd party o utilizando la API oficial de Twitter. Por este motivo, se ha añadido un campo *platform* al dataset, derivado a partir del atributo *source*, que indica el dispositivo desde el que se ha realizado cada tweet. Sin embargo, debido a que la cardinalidad del ya mencionado atributo *source* es muy alta, pues contiene un valor diferente para cada tipo de cliente, se ha considerado apropiado agrupar sus posibles valores en cuatro grandes grupos: *Web*: tweets publicados desde la página web oficial, *mobile*: tweets publicados desde dispositivos mobiles o tablets, *API*: tweets publicados a través de la API oficial y *other*: tweets publicados desde otras plataformas (I.e. Tweetdeck, Tweetbot...).

4.– **Crear atributo "crawled_at":** Algunos de los fragmentos del dataset poseen un atributo *crawled_at* en la tabla de perfiles, que indica la fecha en la que los datos de dicho perfil han sido recogidos por los creadores del dataset. Sin embargo, debido a la ausencia de este campo en algunas de las entradas, se ha decidido volver a calcularlo, tomando esta vez como nuevo valor, la fecha del tweet más reciente del usuario en cuestión. Para asegurar la integridad de los datos, se aplicado esta modificación a todas las entradas del dataset, en lugar de únicamente a las que carecían de valor.

5.– **Añadir Clase** / **Clase binaria:** Debido a que el dataset está organizado siguiendo una estructura de directorios, las entradas de este no contienen ningún atributo que indique la clase. Por tanto, se han añadido las siguientes columnas a los ficheros *users.csv*: *class*: indica la clase del perfil (posibles valores: *fake_follower*, *social_spambot*, *trad_spambot*, *human*). *class_bin*: indica la clase del perfil, sin atender al subtipo de bot. (posibles valores: *human*, *bot*).

3.4. Análisis de datos

Esta fase del desarrollo atiende a dos objetivos diferentes. En primer lugar, verificar la integridad de los datos, detectar valores erróneos y determinar si los datos corresponden a una muestra apropiada de la realidad. En segundo lugar, se busca comprender la estructura de los datos e identificar posibles problemas o aproximaciones de cara a la posterior clasificación.

3.4.1. Perfiles

Un buen punto de partida para comenzar a entender el contenido del dataset es observar la distribución de los datos a través de las diferentes clases de perfiles. Se ha elaborado la figura 3.1. En ella, se puede ver cómo, a menudo, los perfiles de tipo *social_spambots* se comportan de forma muy similar a los perfiles genuinos. Sin embargo, por lo general, aun son diferenciables a simple vista. Además, este tipo de figura resulta particularmente útil para detectar valores extraños o outliers.

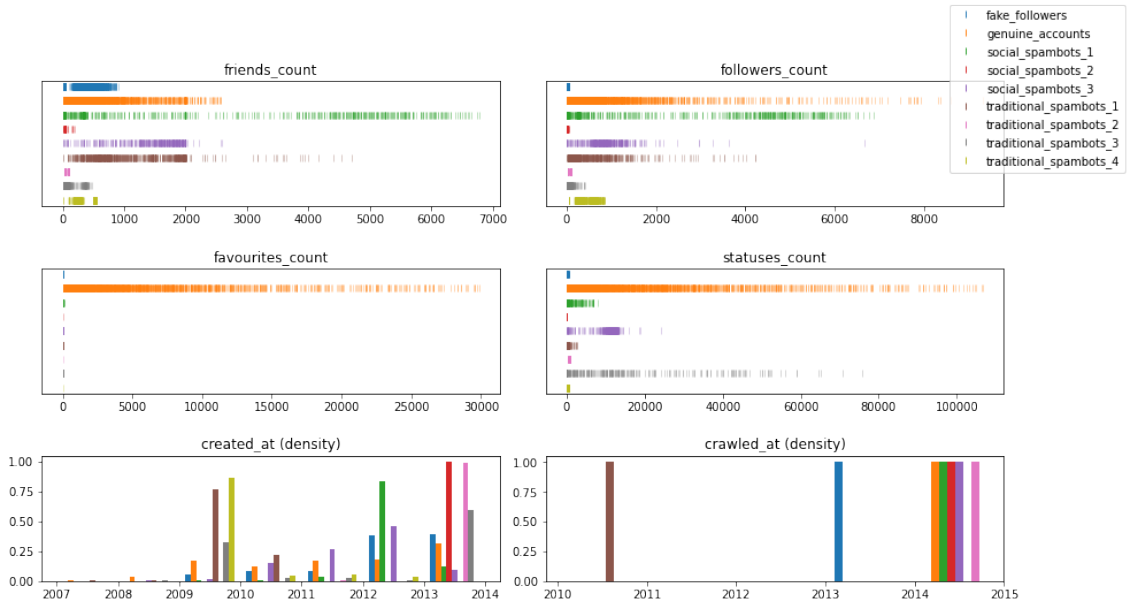


Figura 3.1: Visualización combinada de la distribución de algunos atributos del usuario. Se ejemplifica cómo este tipo de visualización resulta especialmente útil para la detección *outliers* o de errores en el dataset.

Este fenómeno también puede ser comprobado en las figuras 3.2 y 3.3 donde se observa que, en particular el grupo *social_spambots_1* simula bastante bien la relación entre followers y follows de una persona normal, mientras que métodos menos sofisticados tienden a seguir a muchas más cuentas en comparación con las que les siguen.

Además, se ha examinado el rango y el tipo de valores de cada atributo del dataset, con el objetivo de detectar entradas defectuosas o valores atípicos. Algunas gráficas que ejemplifican el proceso

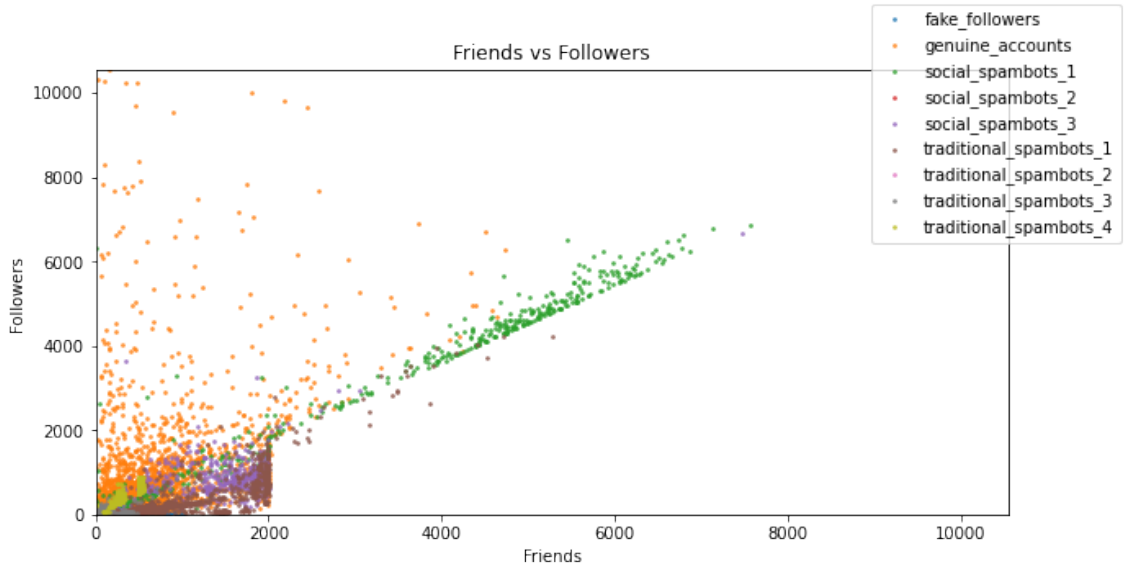


Figura 3.2: Distribución bidimensional de los atributos *friends* y *followers* para las diferentes entradas del dataset. En la figura se aprecian las diferentes clases de perfiles.

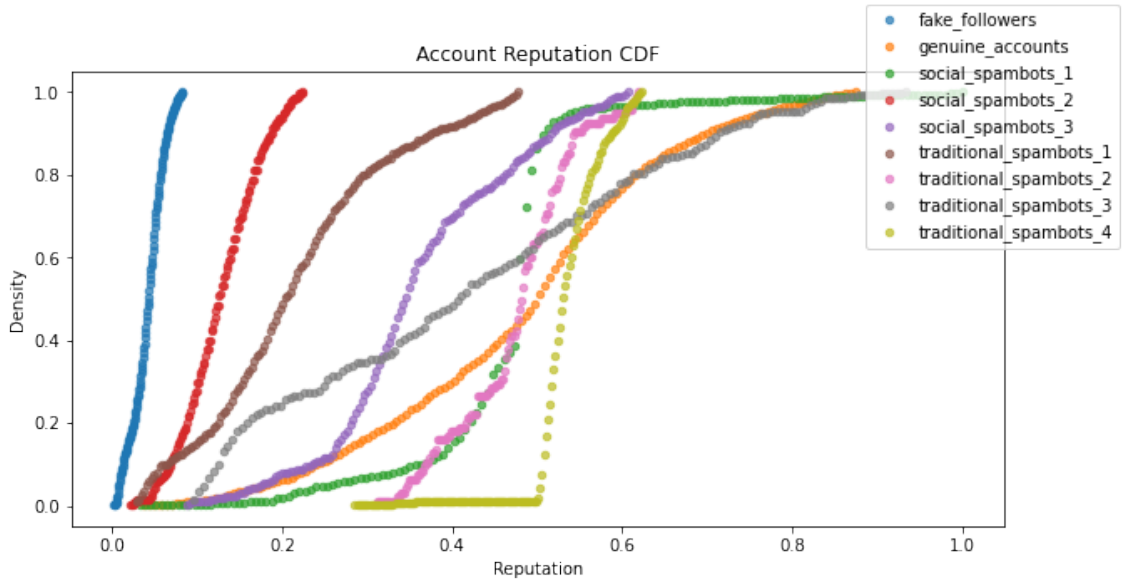


Figura 3.3: Función de densidad acumulada para la reputación de los diferentes usuarios. Ka reputación es calculada como “followers / (followers + friends)”

seguido se encuentran en Análisis de datos, figura A.1.

Por último, es interesante destacar la utilidad de la herramienta *seaborn*, descrita en 2.5, pues nos permite detectar correlaciones en distintos pares de atributos fácilmente. Podemos ver un ejemplo del funcionamiento de esta herramienta en Análisis de datos, figura A.2.

3.4.2. Tweets

Además de emplear técnicas ya comentadas anteriormente para verificar la integridad y correctitud de los datos, es de especial interés visualizar la categoría *platform* añadida anteriormente en Preprocesamiento de los datos. Así, se ha creado un histograma que muestra la cantidad de tweets realizados desde cada plataforma, diferenciando la clase del autor. Así, podemos ver cómo la cantidad de bots que utilizan dispositivos móviles resulta despreciable, en contraste con el número de usuarios genuinos. Por último, cabe destacar, que los bots del tipo *social_spambots* a menudo hacen uso de herramientas de terceros o aparentan funcionar directamente a través de la web.

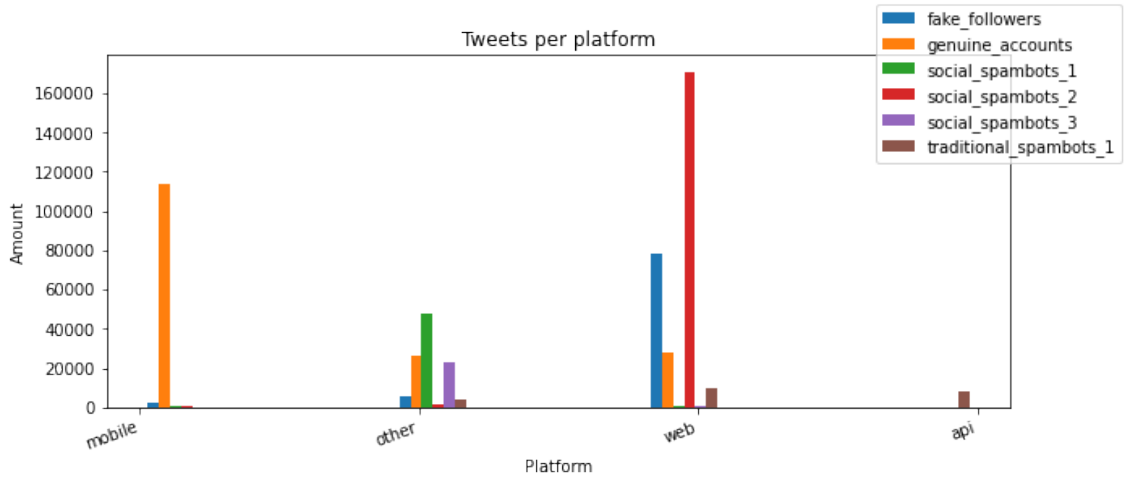


Figura 3.4: Número de tweets realizados por cada tipo de perfil, atendiendo a la plataforma desde la que se emitieron.

3.5. Extracción de características

3.5.1. Extracción de características

Debido a que los algoritmos de aprendizaje automático normalmente trabajan sobre valores numéricos, debemos, en primer lugar, generar una matriz de atributos de este tipo. De cara a generar esta matriz, se ha elaborado una lista de aspectos de interés en la clasificación de perfiles, a partir de otros estudios en la materia [38, 41, 46, 47]. Para cada usuario presente en el dataset, se han computado

y extraído una serie de características, basándose tanto en los datos del dicho usuario, como en sus últimos tweets. Se han propuesto las siguientes categorías y características a extraer en base a las explicadas en otros estudios [5, 15, 19, 24]:

1.– **Account Metadata**: Este primer grupo de atributos se deriva directamente de los datos de perfil que se encuentran en *users.csv*. Contiene información básica sobre las estadísticas del usuario.

   1.1.– **Simple Features**: Estas características son obtenidas directamente a partir de los datos obtenidos mediante la API de twitter, y no requieren ningún tipo de cálculo o procesamiento. Pese a que se espera que los bots más recientes sean capaces de evadir la clasificación mediante este tipo de atributos, todavía pueden ser relevantes en conjunto con otros atributos, o de cara a la detección de bots menos sofisticados.

   1.2.– **Threshold Features**: Estos atributos son de carácter binario. Se derivan a partir de condiciones establecidas basándonos en estudios como [15] y a partir del análisis a priori de los datos del dataset.

   1.3.– **User Growth Features**: Estas características denotan la progresión temporal de los usuarios. Se trata de atributos calculados basándose en la edad de la cuenta de cada usuario.

   1.4.– **Others**: Atributos obtenidos a partir de los datos de la API, pero que requieren algún tipo de cálculo.

2.– **Content Based**: Este grupo contiene atributos derivados del contenido publicado por el usuario, tanto en los tweets que publica como en su propio perfil.

   2.1.– **User Derived Content Features**: En este grupo de atributos suponemos que, debido a que la descripción y el nombre del perfil son elegidos por el usuario, pueden suponer un factor importante a la hora de clasificar cuentas.

   2.2.– **Tweet Derived Content Features**: El contenido de los tweets supone un factor claramente relevante para la distinción de perfiles.

3.– **Behavioural Features (Tweet Based)**: El comportamiento de los usuarios es a priori uno de los factores más determinantes de cara a la clasificación. Sin embargo, a menudo este es uno de los aspectos más complicados de modelar.

Es importante destacar, también, que se ha procurado que prácticamente todas estas características se encuentren, por definición, en un rango entre 0 y 1. Gracias a esto, no será necesario llevar a cabo una fase de normalización adicional, evitando así encontrar un problema con valores no comprendidos en el dataset. Las características propuestas para cada subgrupo se pueden consultar en las tablas 3.3, 3.4 y 3.5.

Se ha creado un módulo llamado *FeatureCollector* en Python, encargado de procesar los datos de cada usuario en combinación con sus tweets. Este módulo permite seleccionar una serie de características a extraer para posteriormente calcular los valores de dichas características para cada entrada del dataset. Además, cabe destacar que se ha puesto especial interés en la facilidad de expansión de esta herramienta, de forma que en un futuro se puedan añadir métodos para extraer nuevos atributos en caso de considerarse necesario. En el código apartado 3.5.1 se incluye un ejemplo básico del uso de este módulo, con el objetivo de ilustrar su funcionamiento.

Nombre	Descripción
1.1 AM - Simple	
has_location	La localización del usuario es pública
has_name	El usuario posee un nombre de perfil
has_description	El usuario posee una descripción biográfica
has_geo_enabled	El usuario ha habilitado la geolocalización
has_default_profile_image	El usuario conserva la imagen de perfil por defecto
has_default_profile	El usuario no ha modificado el tema de su perfil
has_verified	El usuario ha sido verificado en la plataforma
has_protected	El usuario ha establecido su cuenta como protegida
has_url	El usuario ha establecido una URL asociada
1.2 AM - Threshold	
followers_over_30	El usuario sigue a más de 30 personas
tweets_over_50	El usuario posee más de 50 tweets
follows_less_than_2times_followers	El usuario sigue a al menos 2 veces su número de seguidores
age_over_2months	La edad del usuario es superior a 2 meses
default_profile_image_after_2_months	El usuario no ha cambiado el tema de su perfil y tiene al menos 2 meses de edad
1.3 AM - User Growth	
favourite_growth	Número de favoritos del usuario / edad de la cuenta
tweets_growth	Número de tweets del usuario / edad de la cuenta
followers_growth	Número de seguidores del usuario / edad de la cuenta
friends_growth	Número de cuentas seguidas por el usuario / edad de la cuenta
1.4 AM - Others	
reputation	Followers / Friends + Followers

Tabla 3.3: Características de la clase “Account Metadata” extraídas.

Nombre	Descripción
2.1 CF - User Derived	
url_in_description	La biografía del usuario contiene una url
screen_name_length	Longitud del handle del usuario
name_length	Longitud del nombre del usuario
screen_name_has_digits	El handle del usuario contiene dígitos
name_has_digits	El nombre del usuario contiene dígitos
description_length	Longitud de la biografía del usuario
2.2 CF - Tweet Derived	
has_url_over_10perc	Mas del 10 % de los tweets del usuario contienen urls
has_hashtag_over_90perc	Mas del 90 % de los tweets del usuario contienen hashtags
has_mentions_over_30perc	Mas del 30 % de los tweets del usuario contienen menciones
has_url_ratio	Número de urls en tweets / número de tweets
has_hashtag_ratio	Número de hashtags en tweets / número de tweets
has_mentions_ratio	Número de menciones en tweets / número de tweets
number_of_tokens_mean	Número de tokens medio en cada tweet. Los tokens se calculan dividiendo el tweet mediante los signos de puntuación presentes [28].

Tabla 3.4: Características de la clase “Content Based” extraídas.

Nombre	Descripción
3 Behavioural Features	
retweets_ratio	Número de retweets / número de tweets
retweets_over_90perc	El número de retweets / número de tweets es superior al 90 %
has_replied_someone	Alguno de los tweets es una respuesta a otro usuario
mean_time_between_tweets	Tiempo medio entre tweets del usuario
always_uses_same_platform	El usuario siempre twittea desde la misma plataforma
tweeted_mostly_from_platform	Valor numérico correspondiente a la plataforma desde la que twittea mayoritariamente el usuario
tweeted_mostly_from_web	El usuario twittea mayoritariamente desde la web
tweeted_mostly_from_mobile	El usuario twittea mayoritariamente desde un dispositivo movil o tablet
tweeted_mostly_from_api	El usuario twittea mayoritariamente desde la API oficial
tweeted_mostly_from_other	El usuario twittea mayoritariamente desde plataformas no oficiales
last_3_tweet_regular	Los últimos 3 tweets del usuario son regulares temporalmente

Tabla 3.5: Características de la clase “Behavioural” extraídas.

Código 3.1: Ejemplo de uso de FeatureCollector

```
0  from src.FeatureCollector import FeatureCollector
1
2  # Define user features to extract
3  user_features_to_include = [
4      "has_location",
5      "screen_name_length",
6      "name_length",
7      "screen_name_has_digits",
8      "name_has_digits",
9      "description_length",
10     # ...
11 ]
12
13 # Define tweets features to extract
14 tweets_features_to_include = [
15     "mean_time_between_tweets",
16     "has_hashtag_ratio",
17     "has_hashtag_over_90perc",
18     "has_mentions_ratio",
19     "tweeted_mostly_from_platform",
20     # ...
21 ]
22
23 # Create and configure a new FeatureCollector
24 ft = FeatureCollector()
25 ft.configure(user_features_to_include, tweets_features_to_include)
26
27 # Process source datasets and extract features
28 processed_data, features_extracted = ft.process_users(users_dataset, tweets_dataset)
29
30 # Create a new dataset
31 new_dataframe = pd.DataFrame(processed_data, columns=features_extracted)
```

3.5.2. Dataset resultante

Una vez extraídas las características relevantes de los datos, se ha procedido a normalizar los atributos cuyos límites no se encuentran por definición entre 0 y 1 aplicando *minmax_scale* de *SKlearn*. Además, se han verificado los valores de cada una de estas características para cada fragmento de dataset, un ejemplo de esto se puede encontrar en Verificación de características, figura A.3.

De cara a ofrecer una visualización apropiada del nuevo conjunto de datos, se ha hecho uso de la técnica de reducción de dimensionalidad conocida como *Principal Component Analysis*. Pese a que esta técnica es a menudo empleada para aumentar la velocidad del entrenamiento y la clasificación mediante modelos de aprendizaje automático, en este caso es de especial utilidad para mostrar en dos dimensiones una idea aproximada de la distribución de los datos, comprobando así, si estos son diferenciables entre si. Cabe destacar que, para lograr un correcto funcionamiento con esta técnica, ha sido necesario someter los datos a un proceso de estandarización previo, haciendo uso de *StandardScaler* de la librería *SKlearn*. La representación 2D del dataset puede verse en la figura 3.5.

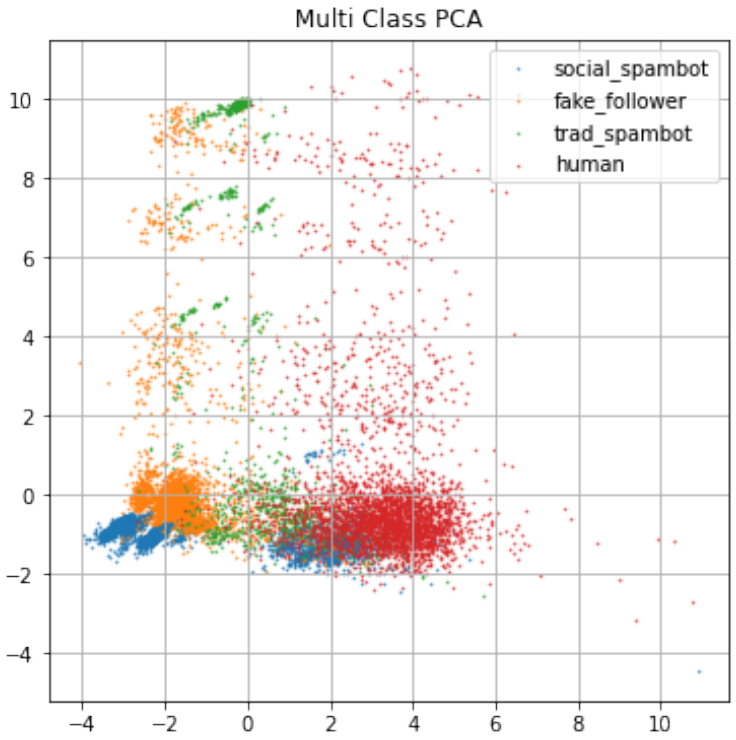


Figura 3.5: Visualización de los usuarios del dataset distinguiendo su clase extraer sus 2 componentes principales mediante PCA

Una vez generados los nuevos conjuntos de valores, se han volcado a nuevos ficheros de cara a la posterior clasificación de estos.

3.6. Clasificación

El objetivo de esta sección es detallar el proceso de comparación de métodos de clasificación aplicados a la información extraída del dataset original.

3.6.1. Métodos de clasificación supervisados

Los métodos de clasificación supervisados suelen proporcionar mejores resultados, a costa de requerir un mayor esfuerzo en el recabado de los datos, debido a que estos han de ser etiquetados, a menudo, manualmente.

Explorar las diferentes posibilidades a la hora de clasificar es altamente importante, de cara a optimizar el proceso de clasificación. Para ello, se ha elaborado una serie de alternativas a probar durante la clasificación. Estas alternativas se exponen a continuación, así como la nomenclatura para cada resultado.

Clasificación Binaria / Clasificación Multi-Clase

Utilizar un modelo binario o multi-clase puede impactar el rendimiento altamente, especialmente en el caso de algoritmos como Random Forest basados en decisión. De esta forma, se llevarán a cabo pruebas diferenciando entre *bot* y *human* (clasificación binaria), y *social_spambot*, *trad_spambot*, *fake_follower* y *human*.

Feature Set

Un aspecto relevante que no solo condiciona la precisión, sino también el tiempo de entrenamiento y clasificación es el número de atributos tomados en cuenta. En algunos casos, realizar una limpieza de atributos poco relevantes puede además mejorar la precisión y reducir el sobreajuste. Por último, si se desea interactuar con la API oficial de Twitter, es preciso tener en cuenta los diferentes contextos a solicitar, pues un elevado número de peticiones puede provocar limitaciones en el acceso a la herramienta [2]. Así, se ha decidido diferenciar entre tres categorías: *all*, todos los atributos, *user*, solo atributos derivados del perfil del usuario y *tweets*, solamente atributos derivados de los tweets publicados por el usuario.

Account Set

Como se ha demostrado en [2], se pueden obtener buenos resultados utilizando clasificadores especializados para cada tipo de bot. Esto además parece resultar especialmente útil de cara a realizar modificaciones o ampliaciones futuras sobre el modelo. Así se ha considerado evaluar cada tipo de bot

de forma individual, así como todos ellos de forma conjunta. Por tanto, se han establecido 4 categorías: *all*, *social_spambots*, *trad_spambots*, *fake_followers*.

Clasificador e Hiperparámetros

Quizás la parte más importante a la hora de construir un modelo es la elección de un algoritmo de clasificación apropiado. Además, la gran mayoría de clasificadores admiten una serie de hiperparámetros, que permiten al usuario influir en el comportamiento de dicho algoritmo, pudiendo adecuarlo así al problema en cuestión.

Tras considerar las diferentes técnicas de ajuste de hiperparámetros existentes, se ha decidido optar por *GridSearchCV*. Se trata de una implementación de la librería *Scikit-learn* que además conlleva una fase de verificación de resultados mediante validación cruzada de cara a evitar el posible sobreajuste. En concreto, cabe destacar que se ha decidido emplear 3 *folds* en este proceso.

En cuanto a los clasificadores en si, resta añadir que, pese a que existe multitud de variantes, hemos seleccionado un conjunto de clasificadores a probar, observando la naturaleza del problema y basándonos en otros estudios similares [2, 15, 25, 41]. Además, se ha tenido en cuenta contar con alta variedad en cuanto al funcionamiento interno de los algoritmos de clasificación. Los algoritmos elegidos se detallan más adelante en Métodos supervisados escogidos.

Mejores atributos

Como ya se ha mencionado anteriormente, la dimensionalidad de los datos es un factor altamente relevante en el rendimiento de muchos algoritmos de clasificación. Es por eso, que se ha decidido llevar a cabo una segunda etapa de clasificación, empleando solo las 10 mejores características para cada *Feature Set*. En definitiva, se ha creado una distinción más entre *all* (todos los atributos) y *best* (solamente los 10 mejores atributos).

Se consideró determinar la importancia de los atributos mediante el método SHAP [29], sin embargo, debido a era demasiado lento, se acabó optando por basarse en las *feature importances* del mejor estimador de cada clasificación realizada mediante Random Forest.

3.6.2. Métodos supervisados escogidos

Basándonos en los estudios mencionados en 3.6.1, hemos elaborado la siguiente selección de métodos a comparar.

- **Random Forest:** Suele ser capaz de ignorar *outliers*, debido a la naturaleza local del modelo. Además, podemos usarlo para derivar una estimación de la importancia de cada atributo en el conjunto del dataset. Los hiperparámetros seccionados para este clasificador se pueden ver en la tabla 3.6(a).
- **Support Vector Machines:** Se trata de un algoritmo robusto que suele funcionar bien incluso si los datos no son linealmente separables, en general, supone un buen punto de partida y es un modelo usado muy a menudo con todo tipo de fines. Los hiperparámetros seccionados para este clasificador se pueden ver en la tabla 3.6(b).
- **Regresión Logística:** Este método tiende a proporcionar buenos resultados cuando los datos son linealmente separables, además normalmente suele ser uno de los algoritmos más rápidos de entrenar. Los hiperparámetros seccionados para este clasificador se pueden ver en la tabla 3.6(c).
- **Perceptrón Multi-capa:** Es un tipo muy flexible de clasificador que suele ofrecer buenos resultados con cualquier tipo de datos. Los hiperparámetros seccionados para este clasificador se pueden ver en la tabla 3.6(d).

3.6.3. Métodos de clasificación no supervisados

Este tipo de métodos normalmente proporcionan un peor rendimiento en términos de precisión, pues logran clasificaciones menos específicas, sin embargo, siguen siendo capaces de alcanzar resultados aceptables, especialmente si se presta atención a la selección de características. Son a menudo usados debido a que no requieren etiquetado de datos, por lo que pueden procesar fácilmente una gran cantidad de estos.

Kmeans

En [15] se sugiere el uso de algoritmos de clustering no supervisado para la clasificación de perfiles falsos en Twitter. Además, existen algunos estudios que reportan resultados exitosos con esta metodología [9]. Por tanto, siguiendo tal recomendación se ha decidido emplear un clasificador basado en KMeans, dado que se trata de una técnica ampliamente conocida y utilizada en el campo del aprendizaje automático. De esta forma KMeans supone un buen punto de partida para comprobar la eficacia de los algoritmos de clustering en este tipo de problemas. Dado que KMeans no es un algoritmo de clasificación en sí, no puede asignar etiquetas a los datos que analiza, simplemente determinar agrupaciones entre los mismos. Los hiperparámetros seccionados para este clasificador se pueden ver en la tabla 3.7.

Sin embargo, existen modelos mixtos capaces de etiquetar los diferentes clusters tras el entrenamiento. En nuestro caso, debido a que poseemos datos etiquetados, se ha decidido emplear una técnica similar a *K Nearest Neighbours*, de forma que, para cada cluster, se tomarán los K puntos más cercanos en el set de entrenamiento y se asignará la clase mayoritaria a dicho cluster. Otras al-

Random Forest (RF)

Parámetro	Valores
bootstrap	True
max_depth	80, 90, 100
max_features	2, 3
min_samples_leaf	3, 4, 5
min_samples_split	8, 10, 12
n_estimators	100, 200

(a) Random Forest

Support Vector Machines

Parámetro	Valores
C	0.1, 1
gamma	0.1, 1
kernel	'rbf', 'sigmoid

(b) Support Vector Machines

Regresión Logística (LogReg)

Parámetro	Valores
penalty	'l1', 'l2'
C	np.logspace(0, 4, 10)

(c) Regresión Logística

Multi-layer Perceptron (MLP)

Parámetro	Valores
hidden_layer_sizes	(10,30,10), (20,)
activation	'tanh', 'relu'
solver	'sgd', 'adam'
alpha	0.0001, 0.05
learning_rate	'constant', 'adaptive'

(d) Multi-layer Perceptron

Tabla 3.6: Estas tablas muestran los hiperparámetros empleados en la clasificación con algoritmos supervisados.

ternativas pueden ser: tomar la clase del punto más cercano al cluster, o tomar la clase mayoritaria entre todos los puntos del cluster. Este modelo mixto nos permite medir el rendimiento de el algoritmo del cluster en términos de precisión, cosa que sin conocer las etiquetas de cada cluster habría sido imposible.

Para realizar una clasificación efectiva mediante este algoritmo, es recomendable someter los datos a PCA con antelación, pues esto facilitará la visualización y el rendimiento. Para ello, se han estandarizado los datos mediante *StandardScaler* de *SKlearn*. Otra alternativa que por lo general produce peores resultados ha sido emplear las 3 características más representativas del dataset.

Por último, cabe añadir que existen métricas específicas que permiten comparar algoritmos de clustering como V-Measure, sugerida en [39]. Sin embargo, se ha optado por *Silhouette coefficient*, *Davies–Bouldin index* y *Calinski Harabasz*, explicadas en 2, debido a que no requieren etiquetado basado en clusters, algo de lo que no disponemos.

Random Forest (RF)

Parámetro	Valores
n_clusters	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

(a) Kmeans

Tabla 3.7: Esta tabla muestra los hiperparámetros empleados en la clasificación con algoritmos no supervisados.

Nomenclatura de resultados

A fin de facilitar la comparativa entre todos los métodos de clasificación mencionados, se ha establecido una determinada notación para estructurar los resultados obtenidos. Dicha notación se explica a continuación:

<feature_amount>-<featureset>-<class_type>-<accountset>-<classifier>

- **feature_amount**: *all*, *best*.
- **featureset**: *all*, *user*, *tweets*.
- **class_type**: *y*, *y_bin*.
- **accountset**: *all*, *fake_followers*, *social_spambots*, *trad_spambots*.
- **classifier**: *RF*, *LogReg*, *SVM*, *MLP*, *KM*.

4

Resultados

Los resultados obtenidos encajan con los publicados por estudios similares tales como [15, 24]. A continuación, se exponen los resultados obtenidos tras las pruebas descritas en Diseño y Desarrollo. Para comprender la notación de los resultados se recomienda referirse a Nomenclatura de resultados. Se ha considerado apropiado dividir esta sección de acuerdo con el carácter de los modelos empleados. Se hará referencia a los resultados completos de la clasificación, que pueden ser encontrados en Resultados de modelos supervisados y Resultados de modelos no supervisados.

4.1. Modelos supervisados (Dataset Completo)

4.1.1. Precisión

Los resultados de los modelos supervisados entrenados con un con el set completo de atributos presentan una precisión muy alta, incluso a la hora de reconocer las diferentes categorías de bots (clase no binaria). Sin embargo, no debemos olvidar que estamos comparando precisión su precisión de forma absoluta, por lo que entrenar modelos especializados para cada tipo de bot continúa siendo una mejor alternativa. Además, esto último también presenta ventajas desde el punto de la escalabilidad [41].

En concreto destaca en este aspecto el algoritmo **Random Forest**, que logra buenos resultados en casi todas las pruebas. Por otra parte, **MLP** también ha probado una alta eficacia, especialmente en modelos entrenados solamente con datos de usuario o de tweets (*featureset*), y dependiendo de la situación es posible que logre un menor overfitting.

Al emplear datos solamente limitados a los usuarios, se obtiene un rendimiento considerablemente peor clasificando social_spambots. Por el contrario, se obtiene un rendimiento excelente clasificando este tipo de bots cuando se emplean datos obtenidos solamente a partir de los tweets. Por tanto, inferimos que emplear un clasificador especializado para este tipo de bots, que sea entrenado con métricas relacionadas con el comportamiento y el contenido de los tweets es una buena aproximación.

4.1.2. Tiempo de entrenamiento

En lo referente a tiempos, de entrenamiento, podemos comprobar cómo, el algoritmo **LogReg** es bastante más rápido que los demás modelos, por tanto puede suponer una buena alternativa a RF en caso de emplear el featureset completo (featureset: “all”) para el entrenamiento.

Por otra parte, cabe destacar que **MLP** tomó mucho tiempo de entrenamiento, especialmente cuando se utilizó el accountset completo (accountset: “all”). No hay que olvidar, sin embargo, que esto se debe a que el número de entradas en los datos es superior que al entrenar modelos independientes para cada tipo de bot. Sin embargo, aun con esto, se puede comprobar claramente cómo MLP sigue siendo significante mente más lento que las otras alternativas.

Finalmente cabe añadir que el rendimiento de **SVM** varía dependiendo del accountset utilizado, pero de forma general es inferior a los obtenidos mediante Random Forest.

4.1.3. Tiempo de clasificación

El tiempo de clasificación ha resultado ser similar entre los diferentes modelos y supone un aspecto despreciable en la mayoría de los casos. Sin embargo, no es así cuando hablamos del algoritmo **SVM**, que requiere significativamente mayor tiempo en los modelos entrenados con social_spambots y accountset “all”’. Esto se vuelve especialmente acusado cuando tratamos con múltiples clases (clase no binaria) y sobre todo para los modelos entrenados usando solo datos de usuario.

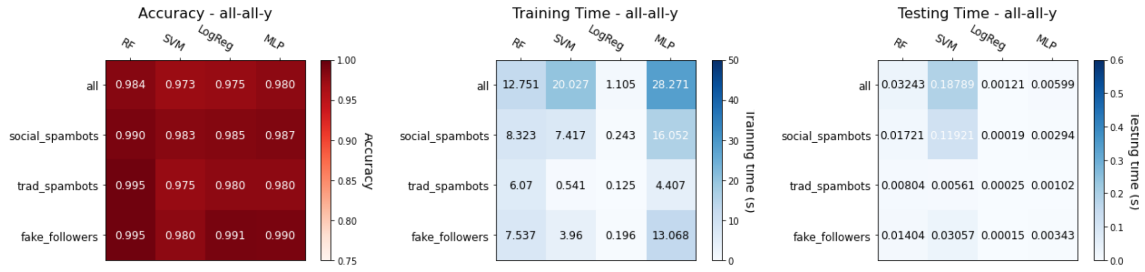


Figura 4.1: Se muestran los resultados obtenidos con los modelos supervisados, entrenado con el featureset “all”. De izquierda a derecha: Precisión, Tiempo de Entrenamiento y Tiempo de Clasificación

Los resultados completos pueden ser encontrados en Resultados de modelos supervisados.

4.2. Modelos supervisados (Mejores 10 atributos)

4.2.1. Precisión

Podemos observar que el uso de modelos entrenados con una selección reducida de 10 atributos ("best") genera resultados claramente inferiores en precisión en comparación con los clasificadores entrenados con el conjunto de atributos completo. A pesar de esto, **RF** parece continuar ofreciendo un rendimiento similar independientemente del featureset elegido. Es probable que un método alternativo de selección de atributos permita generar resultados similares a los modelos entrenados con el featureset "all" para el resto de los clasificadores (especialmente MLP), así que no deberíamos descartar esta opción.

Por otra parte, cabe destacar que el accountset social_spambots resulta en general muy difícil de clasificar al emplear una selección de los atributos más relevantes, especialmente en el caso de utilizar etiquetas no binarias.

Cabe destacar como, en esta batería de pruebas, la clasificación usando el accountset "all" y etiquetas no binarias ("y") resulta muy poco precisa. Esto sugiere que los 10 atributos escogidos como más relevantes no suponen una distinción suficiente entre los tipos de bots presentes. La solución, por tanto, pasa por refinar la metodología de clasificación, recalculando los 10 atributos más relevantes para cada clase de bot o utilizando métodos alternativos a *RF feature importance*. Aun así, dado que esto sucede al mismo tiempo para todos los modelos, es posible que se deba a los conjuntos generados para entrenamiento y clasificación.

4.2.2. Tiempos de entrenamiento

Podemos observar que, al emplear el set de solamente 10 atributos, el tiempo de entrenamiento de los diferentes algoritmos varia ligeramente, en concreto:

- **SVM:** reduce ampliamente el tiempo empleado, a costa como ya se ha mencionado antes de precisión.
- **RF:** aumenta su tiempo de entrenamiento, especialmente al emplear el featureset "users" y clase no binaria "y".
- **MLP:** mejora ligeramente su rendimiento temporal en la mayoría de las pruebas.

4.2.3. Tiempos de clasificación

Al reducir el número de atributos, también, comprobamos como los tiempos de clasificación varían ligeramente con respecto al uso del dataset completo. Cabe destacar especialmente el caso de **SVM**, pues sus resultados no parecen ser homogéneos. Esto se puede porque al entrenar con "best_users_y", el modelo ofrece tiempos de clasificación considerablemente más lentos de lo normal, y, por otra parte, al entrenar con "best_users_y" ofrece resultados considerablemente mejores. Así,

entendemos que el tiempo de clasificación no se debe tanto a la dimensionalidad del dataset, sino más bien al conjunto de datos escogido en cada ejecución.

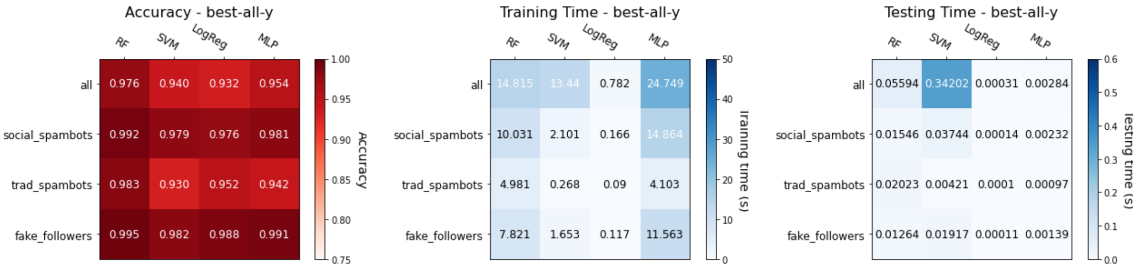


Figura 4.2: Se muestran los resultados obtenidos con los modelos supervisados, entrenado con el featureset "all" y usando solamente los 10 mejores atributos. De izquierda a derecha: Precisión, Tiempo de Entrenamiento y Tiempo de Clasificación

Los resultados completos pueden ser encontrados en Resultados de modelos supervisados.

4.3. Modelos no supervisados

4.3.1. Precisión

Antes de comentar los resultados de **KMeans**, debemos recordar que, de forma previa, se ha sometido al dataset completo a estandarización y PCA con 3 componentes. Además, cabe destacar **la importancia de la escala de colores en las gráficas**, que no es idéntica a la empleada en los modelos supervisados.

En general, podemos observar como la clasificación mediante este tipo de clustering ofrece resultados ligeramente inferiores a los obtenidos mediante modelos supervisados. En primer lugar, observamos como la clasificación de "social_spambots" resulta menos efectiva al emplear el featureset "users". Sin embargo, al usar datos obtenidos solamente de los tweets (featureset: "tweets"), vemos como se alcanza un resultado bastante similar al del featureset *all*.

En general, parece que la clasificación de "trad_spambots" no resulta muy efectiva mediante este método, así que deberá refinarse la selección de características, o simplemente emplear un modelo supervisado en su lugar.

Por último, cabe destacar que los perfiles *fake_followers* son fácilmente distinguibles usando modelos entrenados solamente con los datos de usuario (featureset: "users").

4.3.2. Tiempo de ejecución

En cuanto a tiempo de entrenamiento, observamos como **KMeans** ofrece normalmente valores más o menos homogéneos, esto probablemente se deba a que siempre utilizamos un total de 3 atributos correspondientes a los 3 componentes principales del dataset. Sin embargo, en el caso de la clasificación, el tiempo varía ampliamente en función del número de clusters con el que se alcance el mejor resultado (ajuste de hiperparámetros). Optimizar exhaustivamente este parámetro podrá reportar grandes mejoras en la velocidad del algoritmo.

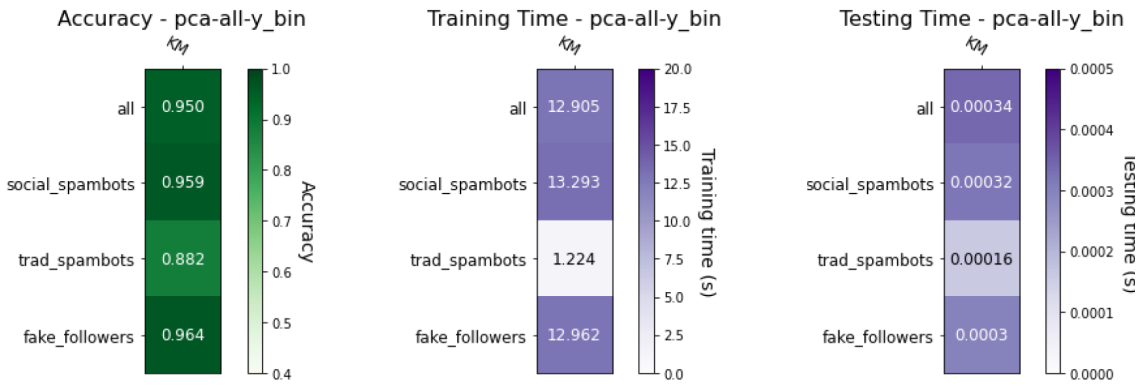


Figura 4.3: Se muestran los resultados obtenidos con los modelos no supervisados, entrenado con el featureset “all”. De izquierda a derecha: Precisión, Tiempo de Entrenamiento y Tiempo de Clasificación. La escala difiere con respecto a las figuras 4.1 y 4.1

Los resultados completos pueden ser encontrados en Resultados de modelos no supervisados, incluyendo comparaciones entre predicciones y realidad para cada modelo entrenado.

4.4. Resumen y modelos propuestos

Dados estos resultados, creemos que existen diferentes posibles soluciones al problema de la clasificación de bots.

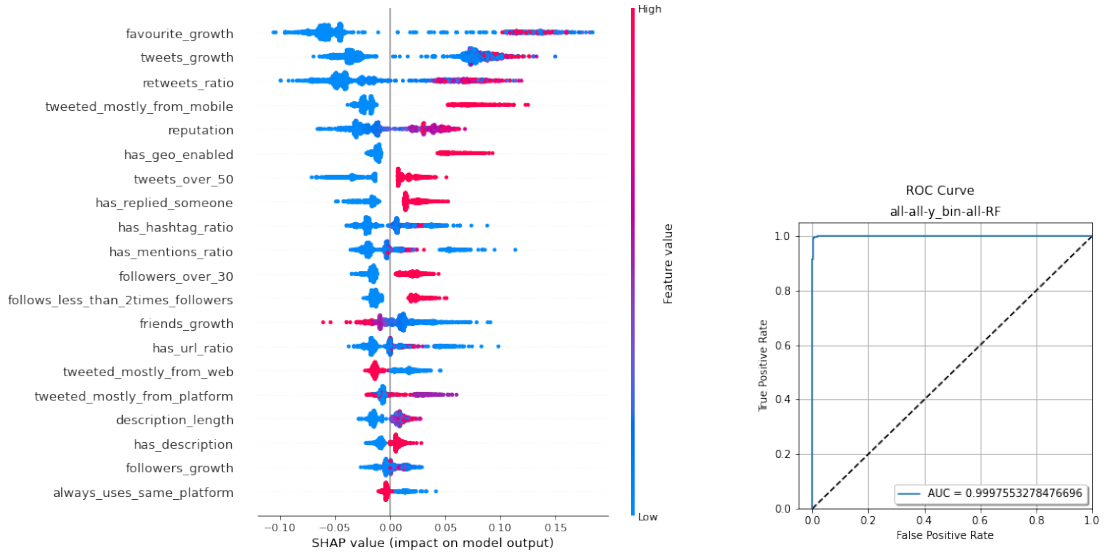
4.4.1. Modelo General

De cara a una clasificación general, claramente la mejor alternativa es el uso de Random Forest haciendo uso de una clase binaria. Las especificaciones de este clasificador se describen a continuación. Asimismo, en la figura 4.4 se muestran la curva ROC de este modelo y un gráfico *SHAP* resumido.

Para una clasificación general se sugiere el siguiente modelo:

1.– **Carácter General** Random Forest: all-all-y_bin-all-RF

1.1.– **C:** 1



(a) Explicación SHAP [29] del mejor modelo general

(b) Curva ROC para el mejor modelo general

Figura 4.4: En la figura se aprecian el gráfico SHAP [29] resumido y la curva ROC correspondientes al mejor modelo de la categoría general **all-all-y_bin-all-RF**.

1.2.– **Gamma:** 0.1

1.3.– **Kernel:** rbf

4.4.2. Modelos especializados

Si bien se requiere un mayor esfuerzo computacional en el entrenamiento y la clasificación, se puede optar por una solución basada en conjuntos de modelos especializados para detectar cada tipo de bot de forma individual. Esto además presenta una gran facilidad de cara a la futura expansión del clasificador, permitiendo así incorporar clasificadores específicos para nuevas clases de bots que puedan surgir.

Se sugieren los siguientes modelos para detectar cada tipo de bot:

1.– **Fake Followers** best-all-y_bin-all-RF

1.1.– **C:** 1

1.2.– **Gamma:** 1

1.3.– **Kernel:** rbf

2.– **Traditional Spambots** all-users-y_bin-trad_spambot-RF

2.1.– **C:** 1

2.2.– **Gamma:** 0.1

2.3.– **Kernel:** rbf

3.– **Social Spambots** best-tweets-y_bin-social_spambot-RF

3.1.– **C:** 1

3.2.– **Gamma:** 1

3.3.– **Kernel:** rbf

4.4.3. Aprendizaje no supervisado

Como ha quedado patente, mediante el uso de técnicas de esta índole es posible alcanzar resultados bastante similares a los obtenidos mediante modelos supervisados. Además, esto presenta la gran ventaja de no requerir el uso de datos etiquetados, eliminando así una gran parte del trabajo realizado de forma manual. Asimismo, se debe considerar la posibilidad de entrenar modelos especializados para cada clase de bot, pues estos han probado ofrecer una mayor precisión que el uso de un único modelo general.

Por otra parte, cabe destacar que, el uso de nuestra implementación de KMeans, pese a reportar una alta precisión, debe considerarse cuidadosamente, pues el tiempo requerido para realizar la clasificación escala exponencialmente con el número de clusters empleados.

Para esta aproximación se sugiere un clasificador mixto con los siguientes modelos:

1.– **Fake Followers** pca-users-y_bin-fake_followers-KM 4.5

1.1.– **N Clusters:** 2

1.2.– **N Neighbours:** 5

2.– **Traditional Spambots** pca-all-y_bin-trad_spambot-KM 4.6

2.1.– **N Clusters:** 14

2.2.– **N Neighbours:** 5

3.– **Social Spambots** pca-tweets-y_bin-social_spambot-KM 4.7

3.1.– **N Clusters:** 9

3.2.– **N Neighbours:** 5

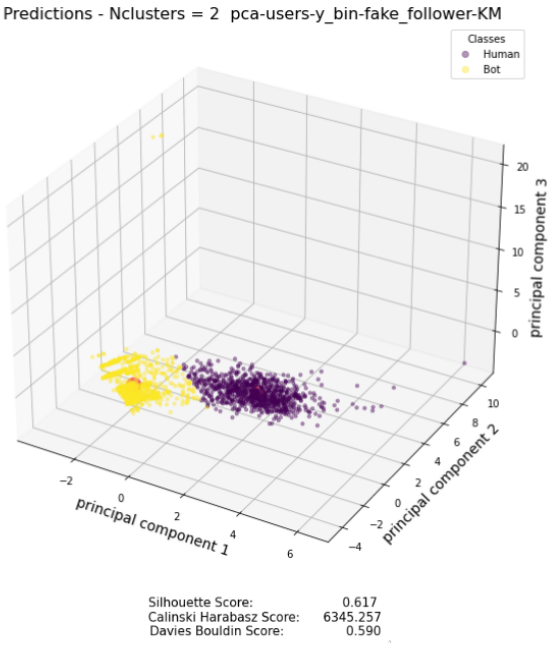


Figura 4.5: Visualización de las predicciones obtenidas por el algoritmo KMeans utilizando el featureset "users" y 2 clusters, para la detección de Fake Followers. Los centros de dichos clusters se representan en color naranja.

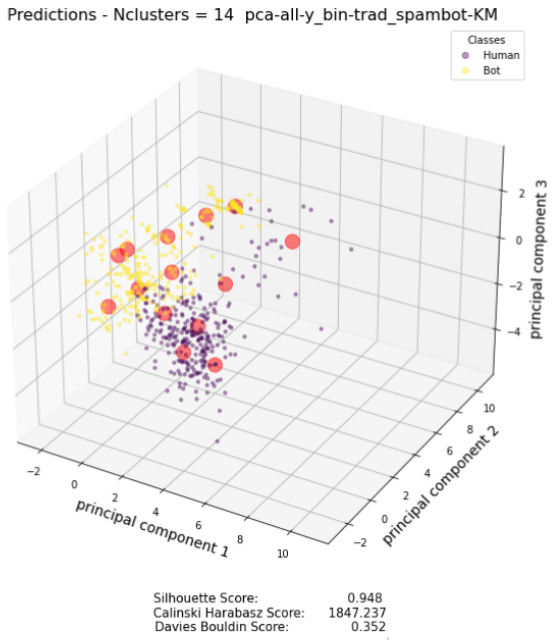


Figura 4.6: Visualización de las predicciones obtenidas por el algoritmo KMeans utilizando el featureset "all" y 14 clusters, para la detección de Traditional Spambots. Los centros de dichos clusters se representan en color naranja.

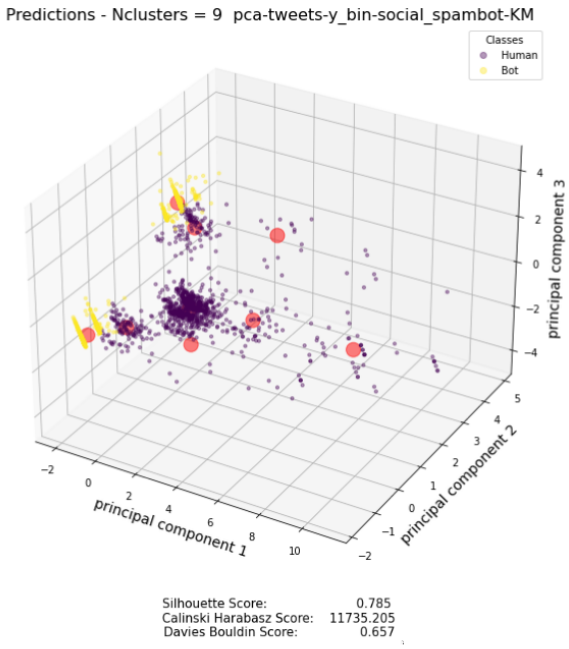


Figura 4.7: Visualización de las predicciones obtenidas por el algoritmo KMeans utilizando el featureset “tweets” y 9 clusters, para la detección de Social Spambots. Los centros de dichos clusters se representan en color naranja.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

Disponer de herramientas consistentes para identificar bots en redes sociales es y seguirá siendo un factor fundamental de cara a reducir el impacto dañino de las *fake news*. Atacar a la fuente difusora de información ha probado ser altamente efectivo, especialmente en comparación con las técnicas centradas en analizar los contenidos de dichas noticias.

Tras la realización de este estudio se ha determinado que el acto de clasificar perfiles de Twitter es un problema relativamente sencillo, sin embargo, la dificultad radica en la obtención de datos no sesgados.

Mediante una comparación exhaustiva de métodos de clasificación y ingeniería de atributos, se establece que Random Forest supone una de las mejores técnicas de clasificación supervisadas, generalmente, independientemente de la selección de características empleada. Asimismo. se ha determinado que, en este caso, determinados tipos de bots resultan más fáciles de clasificar utilizando información extraída de los metadatos del usuario, como es el caso de los *fake_followers*, mientras que otros como los *social_spambots* son distinguibles más fácilmente empleando información derivada de sus tweets.

Por otra parte, ha quedado demostrado como los modelos no supervisados son capaces de obtener resultados similares a los supervisados, eliminando la necesidad de etiquetar el dataset. Sin embargo, la selección de atributos gana más relevancia en estos casos.

Aun con esto, cabe destacar que la evolución constante de los bots y la sofisticación de sus comportamientos, continúa suponiendo un grave inconveniente, pues amenaza con volver obsoletos los modelos estáticos de detección. Por esta razón, los modelos deberán desarrollarse de forma incremental, siendo capaces de admitir extensiones en el futuro.

Por último, cabe añadir que este trabajo ha servido para adquirir una visión generalizada del panorama actual en el ámbito de la clasificación de usuarios en redes sociales, así como diferentes técnicas para visualizar información compleja, y para mejorar el rendimiento de los algoritmos empleados.

5.2. Trabajo Futuro

Pese a que los métodos de detección existentes son capaces de lograr buenos resultados en la actualidad, la constante evolución de los bots supone un complicado desafío. En este aspecto es imperativo prestar atención al desarrollo de nuevas variantes de bots [20, 42]. A su vez, se deberán elaborar métodos de obtención de datos más sofisticados, que garanticen una muestra de perfiles lo menos sesgada posible. Para contar con una mayor facilidad de recabado de datos se sugiere el uso de modelo no supervisados que no requieran interacción humana.

Con el objetivo de lidiar con las ya mencionadas nuevas variantes de bots, se sugiere la elaboración de un modelo de clasificador incremental, con componentes independientes especializados en reconocer tipos particulares de bots. Esta puede ser un área de trabajo prometedora, aunque no se debe olvidar que este tipo de técnicas requerirán un mayor tiempo de entrenamiento. En este aspecto, el correcto tratamiento de los datos y la ingeniería de atributos tomarán especial relevancia [47]. Los diferentes clasificadores pueden ser integrados mediante un modelo de votación que tenga en cuenta el resultado de cada uno de ellos.

Sin embargo, el actual desarrollo de técnicas como el *Deep Learning*, promete complicar la situación actual, por lo que no se deberá dejar de lado el trabajo orientado a la exploración de métodos más complejos de análisis. Un ejemplo de esto puede ser explorar un análisis temporal de las interacciones entre bots tal y como se detalla en [7]. Además, se deberá profundizar en la distinción entre los perfiles automatizados con fines dañinos de los que no [46].

Por otra parte, además de existir una multitud de técnicas de aprendizaje no supervisado diferentes a KMeans, una mayor atención a la selección de características y los conjuntos de entrenamiento puede reportar grandes avances incluso aunque no se varíe el algoritmo [1].

Finalmente, cabe destacar que, una posible vía de estudio en línea con este trabajo sería la adición de otros datasets como [38, 41, 47]. Así como la creación de plataformas similares a *BotOrNot* [17], que permitan a los usuarios comprobar la veracidad de otros perfiles o noticias.

Bibliografía

[1] Arthur, D., and Vassilvitskii, S. K-means++: The advantages of careful seeding. vol. 8, pp. 1027–1035.

[2] Beskow, D. M., and Carley, K. M. Bot-hunter: a tiered approach to detecting & characterizing automated activity on twitter. In *Conference paper. SBP-BRiMS: International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation* (2018), vol. 3, p. 3.

[3] Bessi, A., Coletto, M., Davidescu, G. A., Scala, A., Caldarelli, G., and Quattrociocchi, W. Science vs conspiracy: Collective narratives in the age of misinformation. *PLOS ONE 10*, 2 (02 2015), 1–17.

[4] Brake, C. A machine learning approach to the classification of phishing bot accounts within twitter.

[5] Buntain, C., and Golbeck, J. Automatically identifying fake news in popular twitter threads. *2017 IEEE International Conference on Smart Cloud (SmartCloud)* (Nov 2017).

[6] Caliński, T., and JA, H. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods 3* (01 1974), 1–27.

[7] Chavoshi, N., Hamooni, H., and Mueen, A. Identifying correlated bots in twitter. In *International conference on social informatics* (2016), Springer, pp. 14–21.

[8] Chavoshi, N., Hamooni, H., and Mueen, A. Temporal patterns in bot activities. In *Proceedings of the 26th international conference on world wide web companion* (2017), pp. 1601–1606.

[9] Chen, Z., and Subramanian, D. An unsupervised approach to detect spam campaigns that use botnets on twitter, 2018.

[10] Chino, D. Y. T., Costa, A. F., Traina, A. J. M., and Faloutsos, C. Voltime: Unsupervised anomaly detection on users' online activity volume. In *SDM* (2017).

[11] Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on dependable and secure computing 9*, 6 (2012), 811–824.

[12] Conroy, N. K., Rubin, V. L., and Chen, Y. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology 52*, 1 (2015), 1–4.

[13] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems 80* (Dec 2015), 56–71.

[14] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems 31*, 5 (Sep 2016), 58–64.

[15] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. The paradigm-shift of social spambots. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion* (2017).

[16] Davies, D. L., and Bouldin, D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1*, 2 (1979), 224–227.

[17] Davis, C. A., Varol, O., Ferrara, E., Flammini, A., and Menczer, F. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web* (Republic and Canton of Geneva, CHE, 2016), WWW '16 Companion, International World Wide Web Conferences Steering Committee, p. 273–274.

[18] Del Vicario, M., Bessi, A., Zollo, F., Petroni, F., Scala, A., Caldarelli, G., Stanley, H. E., and Quattrociocchi, W. The spreading of misinformation online. *Proceedings of the National Academy of Sciences 113*, 3 (2016), 554–559.

[19] Ferrara, E., Varol, O., Davis, C., Menczer, F., and Flammini, A. The rise of social bots. *Communications of the ACM 59*, 7 (Jun 2016), 96–104.

[20] Gorwa, R., and Guilbeault, D. Unpacking the social media bot: A typology to guide research and policy: Unpacking the social media bot. *Policy & Internet 12* (08 2018).

[21] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362.

[22] Howard, P. N., and Kollanyi, B. Bots, #strongerin, and #brexit: Computational propaganda during the uk-eu referendum, 2016.

[23] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering 9*, 3 (2007), 90–95.

[24] Knauth, J. Language-agnostic Twitter-bot detection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)* (Varna, Bulgaria, Sept. 2019), INCOMA Ltd., pp. 550–558.

[25] Kosmajac, D., and Keselj, V. Twitter bot detection using diversity measures. In *Proceedings of the 3rd International Conference on Natural Language and Speech Processing* (Trento, Italy, Sept. 2019), Association for Computational Linguistics, pp. 1–8.

[26] Kumar, P. Graph data modeling for political communication on twitter.

[27] L, H. Digital wildfires in a hyperconnected world. wef report 2013, 2013. https://www.aclweb.org/anthology/W19-7401", last accessed June 16, 2020.

[28] Lundberg, J., Nordqvist, J., and Laitinen, M. Towards a language independent twitter bot detector. In *DHN* (2019), pp. 308–319.

[29] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

[30] Minnich, A., Chavoshi, N., Koutra, D., and Mueen, A. Botwalk: Efficient adaptive exploration of twitter bot networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in*

Social Networks Analysis and Mining 2017 (New York, NY, USA, 2017), ASONAM '17, Association for Computing Machinery, p. 467–474.

[31] Mármol, J. P.-G. M. Z. P. N. S. L. B. A. H. C. M. G. P. J. A. R.-V. G. M. P. F. G. Spotting political social bots in twitter: a dataset for the 2019 spanish general election, 2020.

[32] Nickerson, R. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology 2* (06 1998), 175–220.

[33] Oentaryo, R. J., Murdopo, A., Prasetyo, P. K., and Lim, E.-P. On profiling bots in social media. *Social Informatics* (2016), 92–109.

[34] pandas development team, T. pandas-dev/pandas: Pandas, Feb. 2020.

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[36] Pierri, F., and Ceri, S. False news on social media: A data-driven survey, 2020.

[37] Rapoza, K. Can 'fake news' impact the stock market? https://www.forbes.com/sites/kenrapoza/2017/02/26/can-fake-news-impact-the-stock-market/?sh=3320a1e32fac, last accessed June 16, 2020.

[38] Rauchfleisch, A., and Kaiser, J. The false positive problem of automatic bot detection in social science research. *PloS one 15*, 10 (2020), e0241045.

[39] Rosenberg, A., and Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (Prague, Czech Republic, June 2007), Association for Computational Linguistics, pp. 410–420.

[40] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20* (1987), 53–65.

[41] Sayyadiharikandeh, M., Varol, O., Yang, K.-C., Flammini, A., and Menczer, F. Detection of novel social bots by ensembles of specialized classifiers. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Oct 2020).

[42] Stieglitz, S., Brachten, F., Ross, B., and Jung, A.-K. Do social bots dream of electric sheep? a categorisation of social media bot accounts, 2017.

[43] Vargo, C., Guo, L., and Amazeen, M. A. The agenda-setting power of fake news: A big data analysis of the online media landscape from 2014 to 2016. *New Media & Society 20* (2018), 2028 – 2049.

[44] Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software 6*, 60 (2021), 3021.

[45] Wei, F., and Nguyen, U. T. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings, 2020.

[46] Yang, K., Varol, O., Davis, C. A., Ferrara, E., Flammini, A., and Menczer, F. Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies 1*, 1 (Jan

2019), 48–61.

[47] Yang, K.-C., Varol, O., Hui, P.-M., and Menczer, F. Scalable and generalizable social bot detection through data selection. *Proceedings of the AAAI Conference on Artificial Intelligence 34*, 01 (Apr 2020), 1096–1103.

[48] Zhang, J., Zhang, R., Zhang, Y., and Yan, G. The rise of social botnets: Attacks and countermeasures. *IEEE Transactions on Dependable and Secure Computing 15*, 6 (2018), 1068–1082.

Apéndices

A

Análisis de datos

A.0.1. Examen de los atributos del dataset

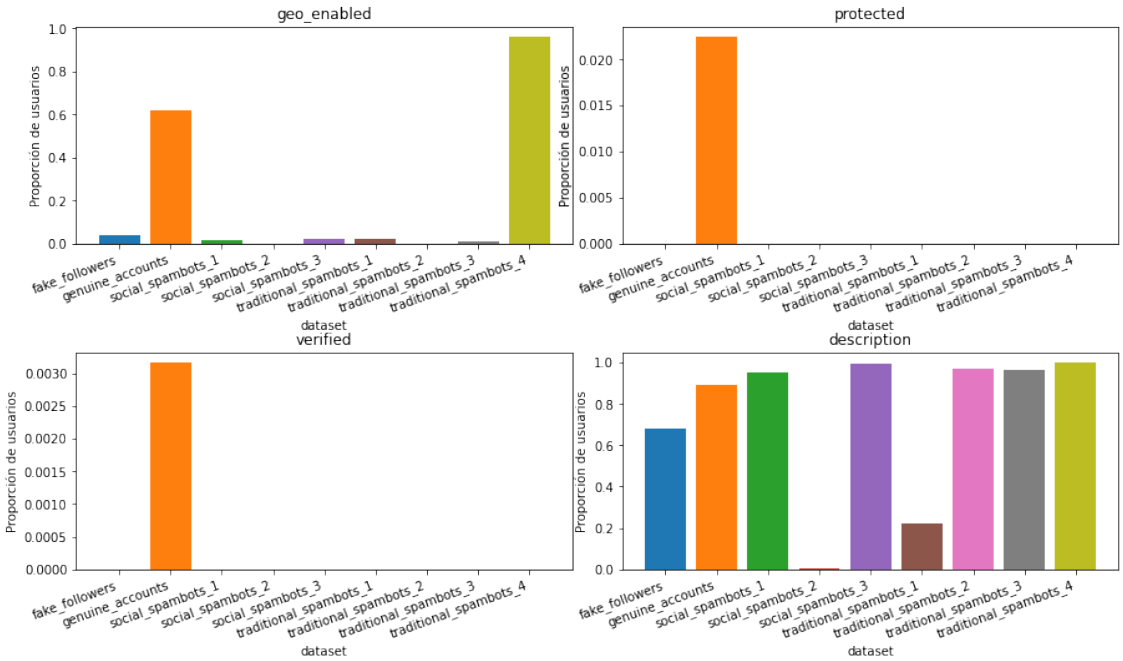


Figura A.1: Ejemplo del método seguido para entender la proporción que siguen algunos de los valores presentes en el dataset. En la figura podemos observar la proporción en la que los valores de determinados atributos son positivos en cada fragmento del dataset.

A.0.2. Correlación entre atributos del dataset

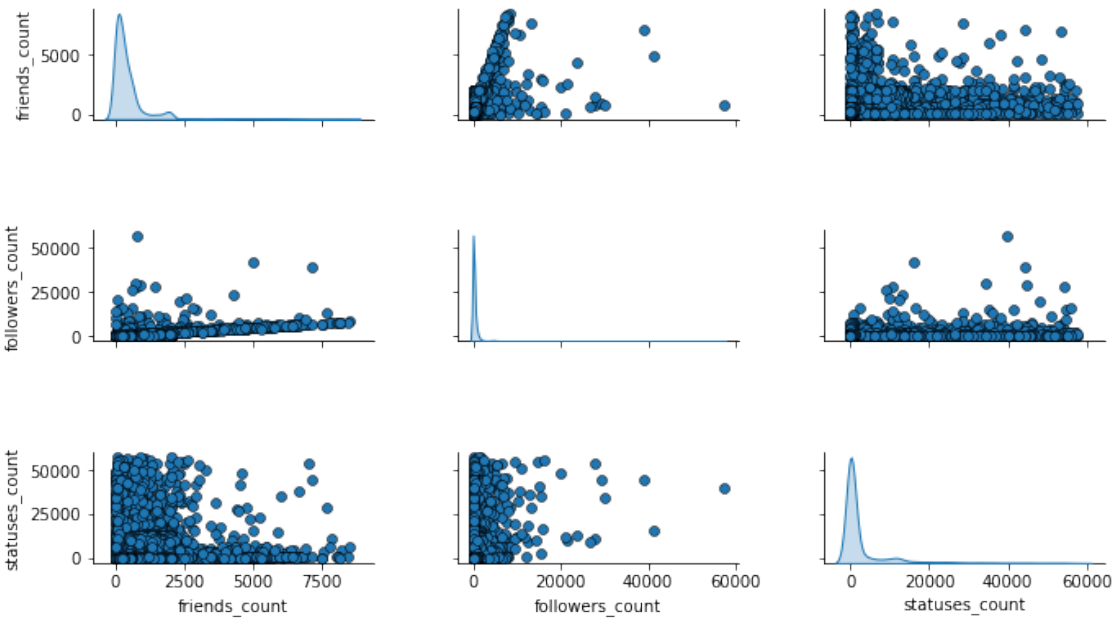


Figura A.2: Ejemplo del método seguido, haciendo uso de la herramienta "Seaborn" para detectar correlación entre distintos atributos del dataset. En la figura podemos observar la distribución y correlación de los atributos *followers_count*, *friends_count* y *statuses_count*.

A.0.3. Verificación de características

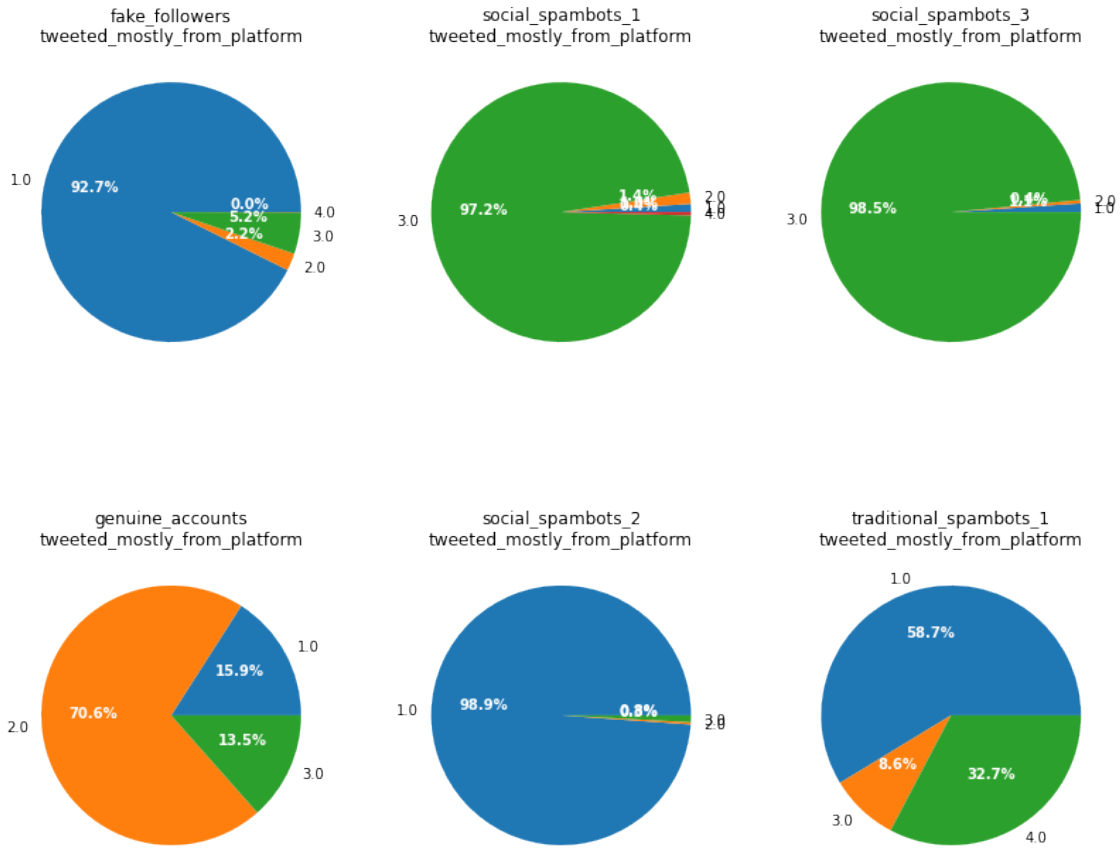


Figura A.3: Ejemplo del método seguido para verificar los valores encontrados en las características extraídas a partir del dataset original.

B

Resultados de modelos supervisados

B.0.1. Resultados: todos los atributos

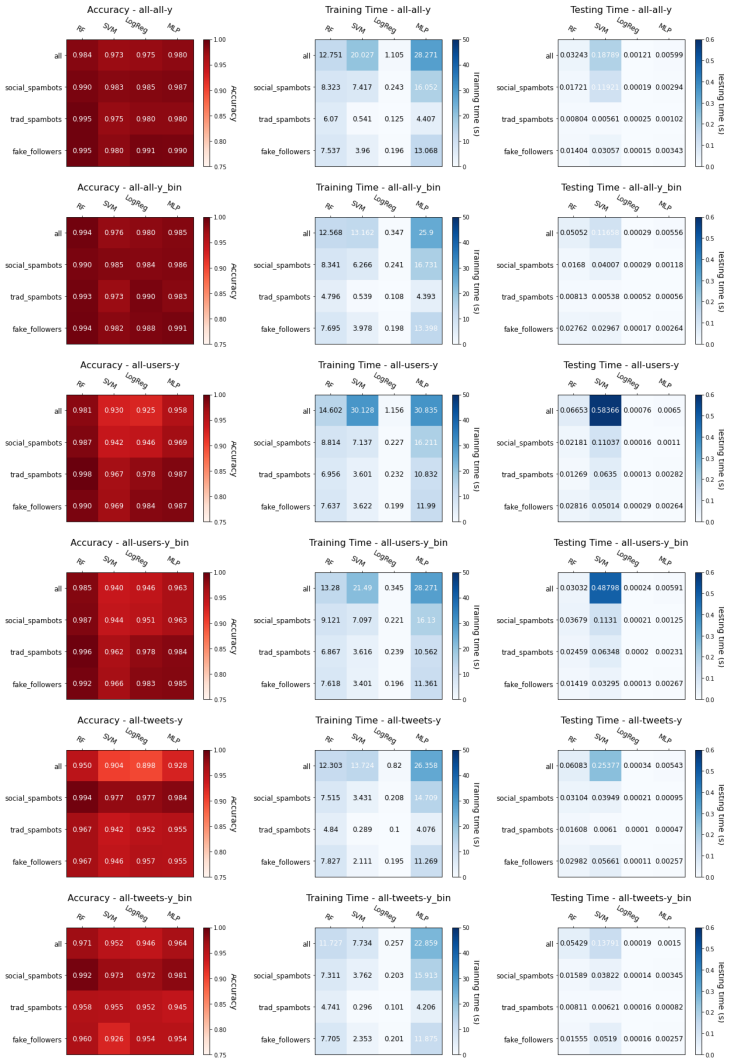


Figura B.1: Resultados obtenidos tras las pruebas de clasificación utilizando modelos supervisados y todos los atributos extraídos del dataset. Los resultados están agrupados por tipo de perfil y clasificador empleado.

B.0.2. Resultados: 10 mejores atributos

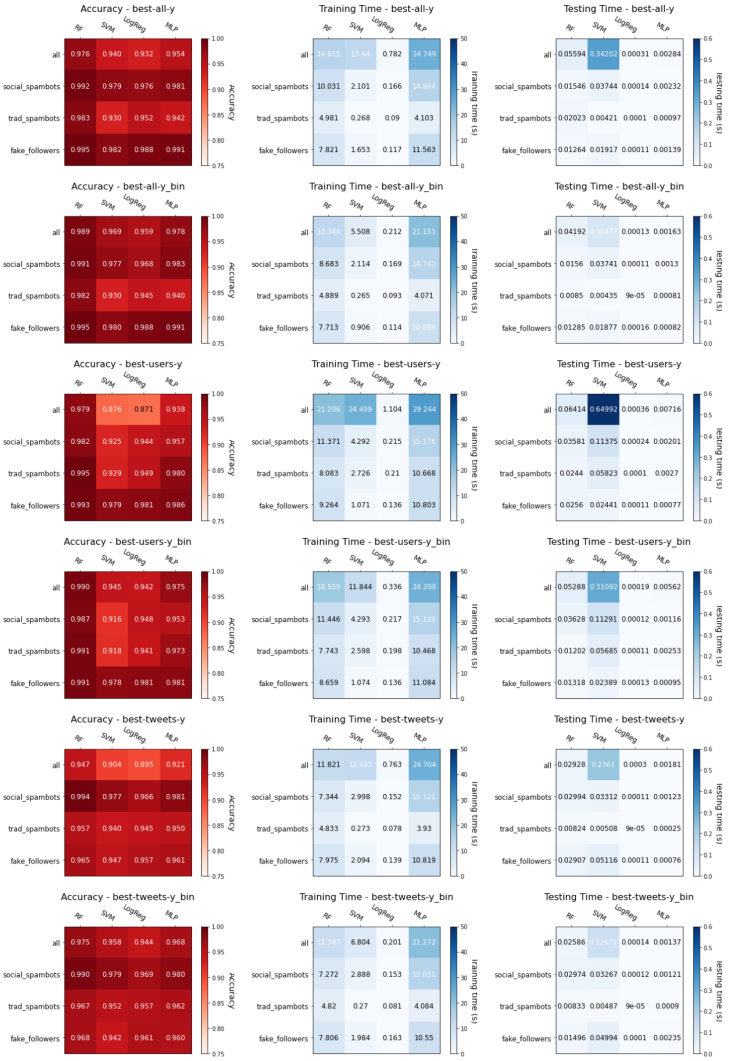


Figura B.2: Resultados obtenidos tras las pruebas de clasificación utilizando modelos supervisados y los atributos extraídos del dataset. Los resultados están agrupados por tipo de perfil y clasificador empleado

C

Resultados de modelos no supervisados

C.0.1. Resultados: tras aplicar PCA

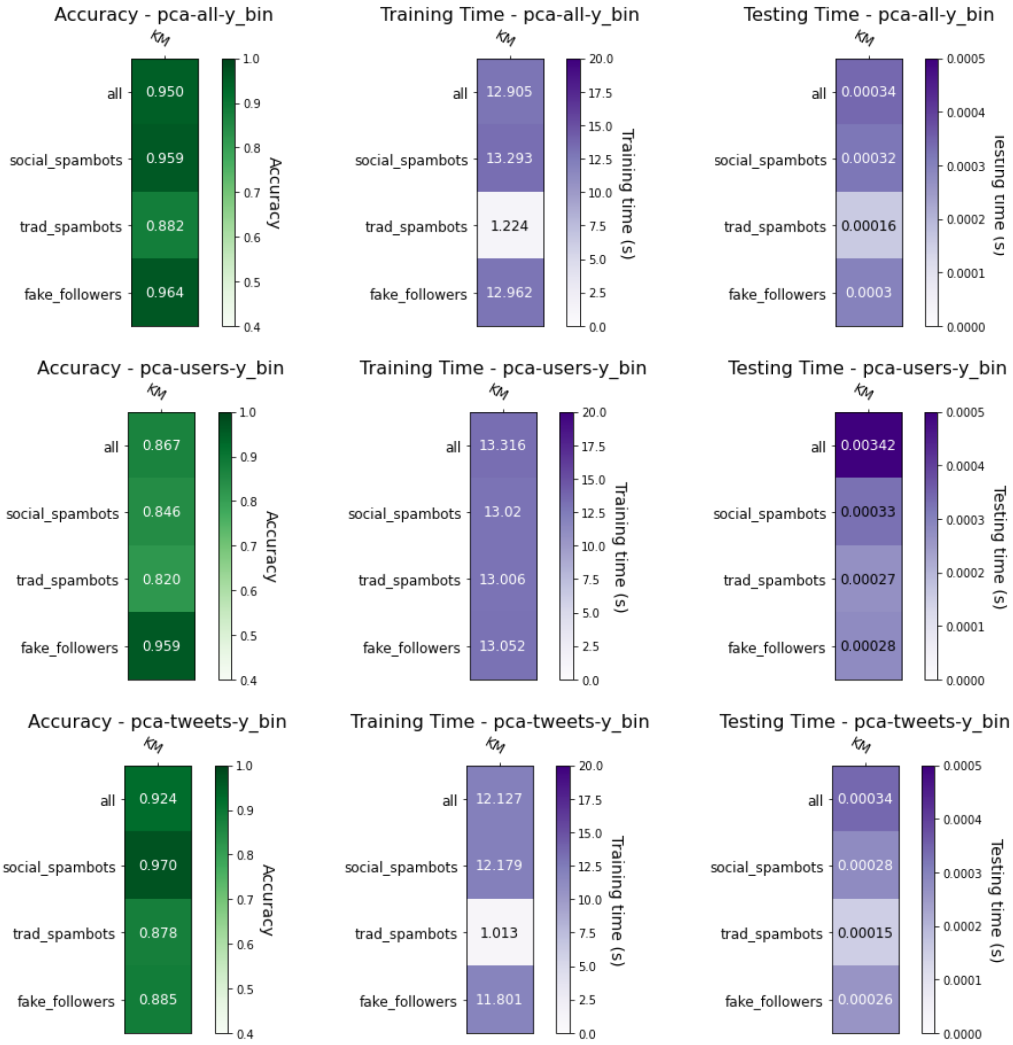


Figura C.1: Resultados obtenidos tras las pruebas de clasificación utilizando métodos no supervisados y pca para reducir la dimensionalidad. Los resultados están agrupados por tipo de perfil y clasificador empleado.

C.0.2. Visualización: usando el featureset “all”

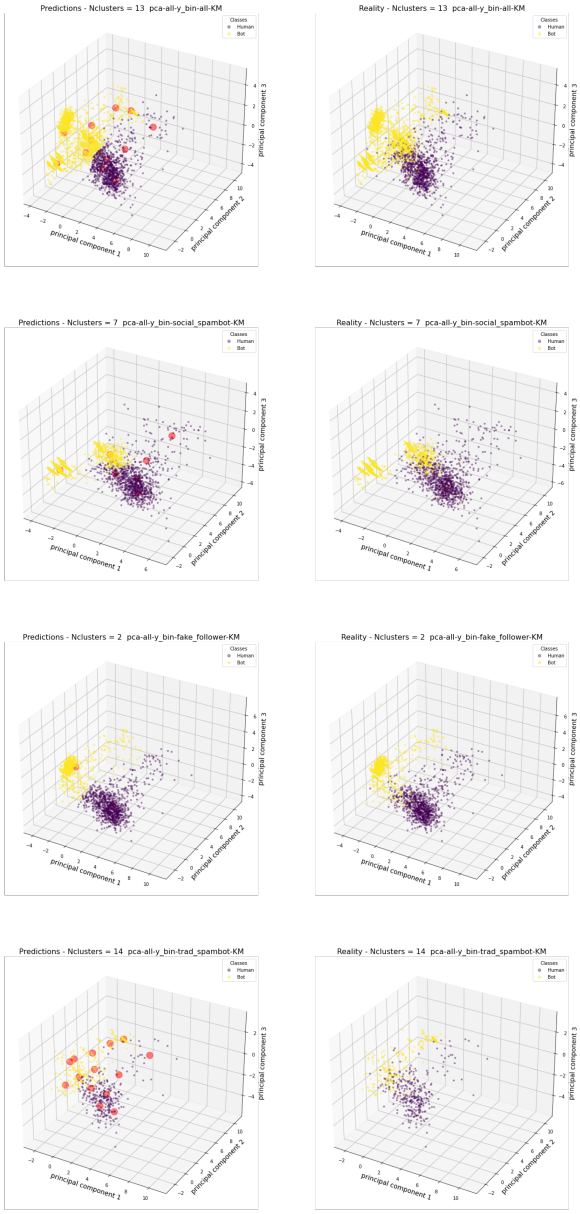


Figura C.2: Visualización de los resultados de clustering con Kmeans usando el featureset “all” y pca para reducir la dimensionalidad. A la derecha se encuentran las predicciones, a la izquierda los datos reales. Los centros de cada cluster se muestran en naranja. Cada cluster ha sido etiquetado con un algoritmo similar a KNN con 5 vecinos.

C.0.3. Visualización: usando el featureset “users”

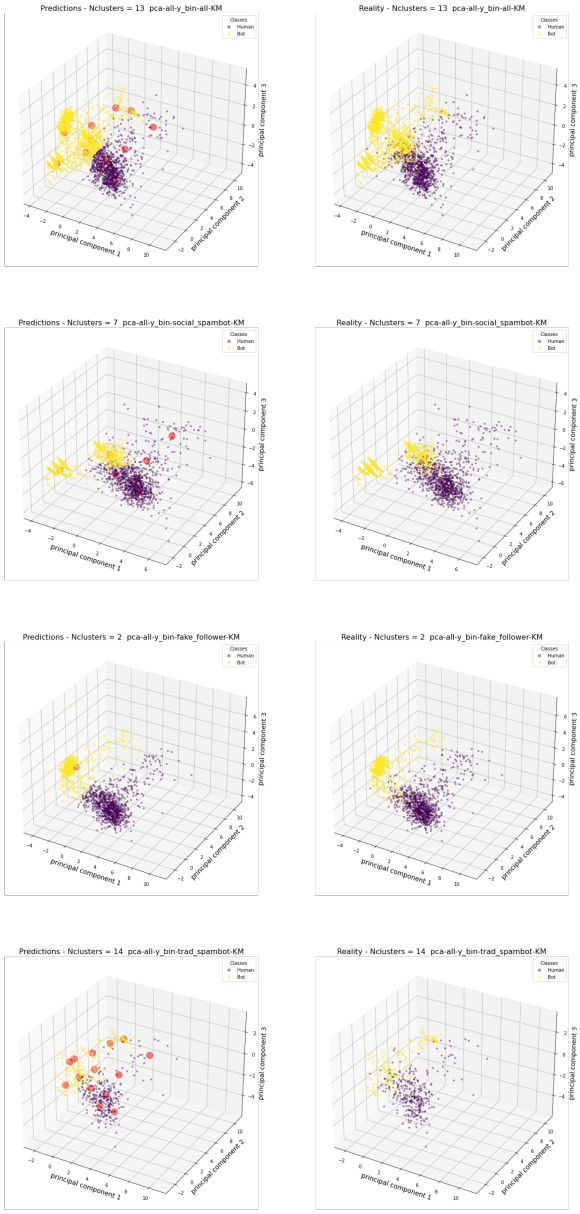


Figura C.3: Visualización de los resultados de clustering con Kmeans usando el featureset “users” y pca para reducir la dimensionalidad. A la derecha se encuentran las predicciones, a la izquierda los datos reales. Los centros de cada cluster se muestran en naranja. Cada cluster ha sido etiquetado con un algoritmo similar a KNN con 5 vecinos.

C.0.4. Visualización: usando el featureset “tweets”

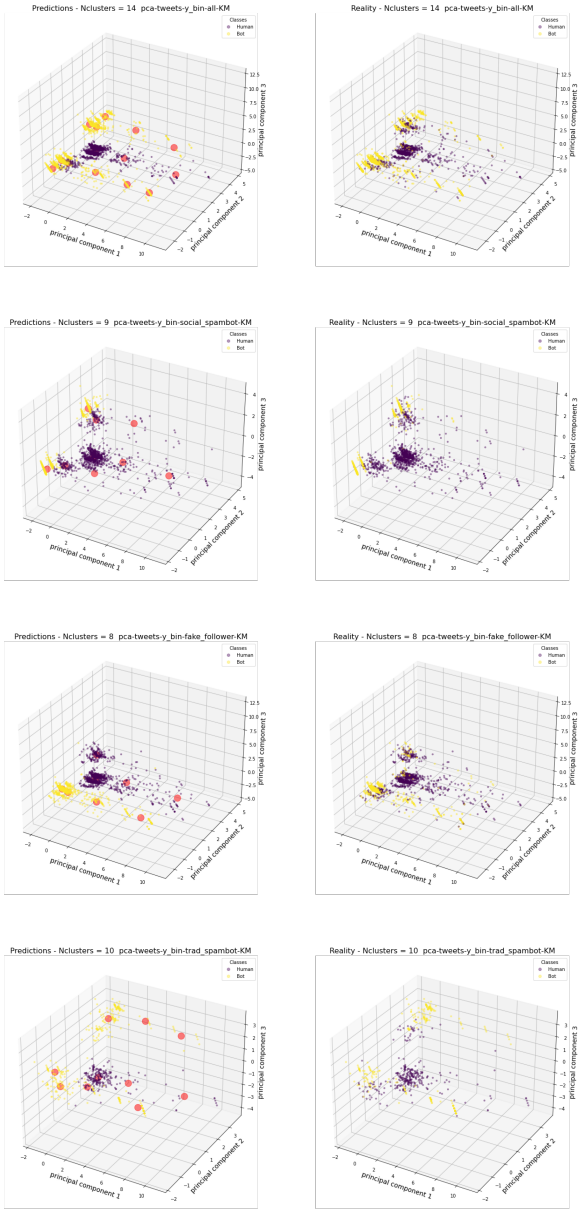


Figura C.4: Visualización de los resultados de clustering con Kmeans usando el featureset “tweets” y pca para reducir la dimensionalidad. A la derecha se encuentran las predicciones, a la izquierda los datos reales. Los centros de cada cluster se muestran en naranja. Cada cluster ha sido etiquetado con un algoritmo similar a KNN con 5 vecinos.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID